all rights reversed

HACKER

Σεπτέμβρης 2012 # Τιμή τεύχους 6,40€ τεύχος 012 - California Dreaming Edition

Απαλλαγείτε από τα ζιζάνια! [*]

Proxy in the Middle attacks

Κάντε τον router σας torrent downloader

Επιθέσεις SQL injection, σχεδόν στα τυφλά

Σχεδιάστε εφαρμογές για το περιβάλλον γραφικών

[*] Ακόμη κι αν το anti-virus που έχετε δεν τα βλέπει

Περιεχόμενα

και οδηγίες χρήσης

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55 133, Καλαμαριά. Η εκτύπωση και η βιβλιοδεσία γίνονται από την ThessPrint A.E. Το email επικοινωνίας του περιοδικού είναι το talk2us@deltahacker.gr και το email για τις συνδρομές το subscriptions@deltahacker.gr. Οι πληροφορίες για τις συνδρομές είναι στο http://deltahacker.gr/subscriptions και οι παραγγελίες γίνονται από το http://deltahacker.gr/order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.


```
(*) Όταν ένα αστέρι πέφτει, ένας μαθηματικός κάνει μια f(x)
[ κλεμμένο από το twitter και συγκεκριμένα από την @marilouJD
-- follow και πείτε ένα *γεια* κι από εμάς! ]
```

Νόμος δύναμης και ξυδάκι

Σκοντάφτουμε πάνω του, σχεδόν παντού: Στη φυσική, στη βιολογία, στη γεωλογία, στις κοινωνικές επιστήμες -- φυσικά και στο Διαδίκτυο. Πρόκειται για έναν πανταχού παρόντα νόμο κι εξετάζοντας τις συνέπειές του βγάζουμε ενδιαφέροντα, ενίοτε και πολύ διασκεδαστικά συμπεράσματα. Αλήθεια σας λέμε.

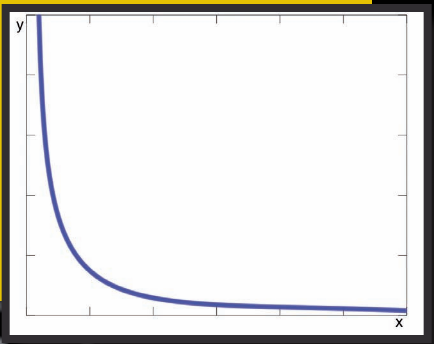
by subZraw

Μιλώντας γενικά, ο νόμος της δύναμης (power law) δεν είναι παρά μια απλή, μαθηματική σχέση μεταξύ δύο ποσοτήτων. Μιλώντας λίγο πιο συγκεκριμένα, όταν η συχνότητα ενός γεγονότος είναι ανάλογη της δύναμης κάποιας μεταβλητής που αφορά στο γεγονός, τότε λέμε ότι η συχνότητα ακολουθεί το νόμο της δύναμης.

Παραμένει γενικός κι αφηρημένος ο ορισμός, ε; Δίκιο έχετε. Μερικά παραδείγματα θα ρίξουν άπλετο φως.

1. Σκεφτείτε τους **σεισμούς** που γίνονται στον πλανήτη. Οι πραγματικά ισχυροί, καταστρεπτικοί σεισμοί, είναι λίγοι. Από την άλλη, όσο πέφτει η ένταση στην κλίμακα ρίχτερ, τόσο συχνότεροι είναι οι σεισμοί της υπό εξέταση κλίμακας.
2. Σκεφτείτε τους **κρατήρες**, στην επιφάνεια της Σελήνης. Οι μεγάλοι κρατήρες σπανίζουν, σε σχέση με τους μικρότερους κρατήρες. Όπως εξηγεί και η κα Σύμβουλος Εκδόσεως (ΤΜ) του περιοδικού, αυτό είναι αναμενόμενο: Η πρόσκρουση ενός μετεωρίτη δημιουργεί έναν μεγάλο κρατήρα και γύρω γύρω μικρότερους. Επίσης, οι προσκρούσεις μεγάλων μετεωριτών είναι σπανιότερες σε σχέση με τις προσκρούσεις μικρότερων.
3. Αναλογιστείτε τους **πολέμους**. Εκείνοι που έχουν μεγάλο αριθμό θυμάτων σπανίζουν, σε σχέση με εκείνους που έχουν μικρότερο αριθμό θυμάτων.
4. Άλλο παράδειγμα, από το χώρο της οικονομίας: Οι πολύ **πλούσιοι** είναι αριθμητικά λιγότεροι από τους λιγότερο πλούσιους ή φτωχούς. Μ’ άλλα λόγια, ο πλούτος έχει μια τάση να συγκεντρώνεται στους λίγους.
5. Ας αλλάξουμε εντελώς κλίμα, περνώντας στο χώρο των ηθοποιών. Οι πολύ **δημοφιλείς ηθοποιοί** είναι αριθμητικά λιγότεροι, σε σύγκριση με τους λιγότερο δημοφιλείς ή πρακτικά άγνωστους.
6. Δείτε επίσης τα blogs (ή τα sites). Παρόμοια συμπεράσματα κι εδώ: Τα πολύ **δημοφιλή blogs** (ή sites) είναι αριθμητικά λιγότερα σε σχέση με τα λιγότερο δημοφιλή ή πρακτικά άγνωστα.

Τι κοινό έχουν άραγε σεισμοί, κρατήρες, πόλεμοι, πλούσιοι, δημοφιλείς ηθοποιοί και δικτυακοί τόποι με μεγάλη επισκεψιμότητα; Μα, η συχνότητα εκδήλωσης ή εμφάνισής τους, ακολουθεί έναν νόμο δύναμης. Παρατηρήστε το γράφημα που παραθέτουμε στο κείμενο. Η εικονιζόμενη καμπύλη είναι της μορφής y = a / xk, όπου ο συντελεστής a και η δύναμη k είναι σταθερές. Με βάση αυτή την καμπύλη, δείτε τώρα πόσο πιο γρήγορα περιγράφουμε τα φαινόμενα των παραδειγμάτων 1 έως 6:



1. x = πλήθος σεισμών, y = ένταση στην κλίμακα Ρίχτερ
2. x = πλήθος κρατήρων, y = τάξη μεγέθους κρατήρα
3. x = πλήθος πολεμικών συγκρούσεων, y = θύματα
4. x = πλήθος ανθρώπων, y = πλούτος
5. x = ηθοποιός, y = δημοτικότητα
6. x = blog (ή site), y = επισκεψιμότητα (π.χ., ανά μήνα)

Είδατε; Πανταχού παρών, ο νόμος της δύναμης! Ισχύει και στον χώρο των χάκερ, φυσικά. Επειδή όμως οι άνθρωποι δεν είμαστε ούτε κρατήρες ούτε νούμερα, υπάρχουν πολλά ενδιαφέροντα που μπορεί να παρατηρήσει κανείς εδώ. Εξηγούμαστε.

Οι δημοφιλείς χάκερ και το καλάμι

Ψάξτε το λίγο και θα δείτε κι εσείς ότι οι δημοφιλείς χάκερ σπανίζουν. Ας το δούμε αναλυτικά αυτό, εκλέγοντας λίγο αυθαίρετα τα νούμερα.

- Στα δάκτυλα του ενός χεριού μετρούνται οι χάκερ για τους οποίους έχουν ακούσει ή διαβάσει έστω και μία φορά *πάνω* από ένα εκατομμύριο άνθρωποι (π.χ., Kevin Mitnick).
- Κάπως περισσότεροι είναι οι χάκερ για τους οποίους έχουν ακούσει ή διαβάσει μερικές εκατοντάδες χιλιάδες άνθρωποι (π.χ., Jeff Moss).
- Πολύ περισσότεροι είναι εκείνοι για τους οποίους έχουν ακούσει ή διαβάσει μερικές δεκάδες χιλιάδες άνθρωποι (π.χ., Dave Kennedy).
- Σχεδόν όλοι οι υπόλοιποι χάκερ είναι γνωστοί μόνο σε κοντινούς φίλους και γνωστούς.

Ίσως διαφωνήσετε με τα νούμερα και τις εκτιμήσεις μας, μάλλον όμως θα συμφωνήσετε με τη γενική ιδέα.

Αίσθησή μας, η οποία μάλιστα έχει τύχει να επιβεβαιωθεί περισσότερες από μία φορές, είναι ότι σε αντίθεση, π.χ., με κάποιους δημοφιλείς ηθοποιούς, οι δημοφιλείς χάκερ δεν έχουν καβαλημένο το καλάμι. Ίσα ίσα που έχουν κοπιάσει πραγματικά για να κερδίσουν αυτή τη δημοφιλία, η οποία μάλιστα ήλθε ως παράπλευρη συνέπεια της διαρκούς προσπάθειάς τους για απόκτηση γνώσεων και προσωπική βελτίωση. Έτσι, αν τύχει να μιλήσετε με κάποιον δημοφιλή χάκερ, το πιθανότερο είναι ότι δεν θα σας πάρει από τα μούτρα, δεν θα σας κοροϊδέψει και ποτέ δεν θα γελάσει περιπαιχτικά με τις όποιες ερωτήσεις σας. Είναι βλέπετε που ξέρει από πρώτο χέρι πόσο δύσκολα σκαρφαλώνει κανείς προς τ' αριστερά της καμπύλης. Γνωρίζει, επίσης, ότι αυτή η ίδια η δημοφιλία δεν μπορεί να είναι αυτοσκοπός.

Το τοπίο αρχίζει ν' αλλάζει όσο κινείται κανείς προς τα δεξιά της καμπύλης. Είναι βλέπετε που αυξάνει δραματικά η πιθανότητα να πέσεις σε "χάκερ", οι οποίοι ενδιαφέρονται περισσότερο να γίνουν πριμαντόνες παρά να μάθουν καλά πέντε έξι πράγματα. Οι ίδιοι βέβαια ισχυρίζονται ότι το κατέχουν το σπορ, η συμπεριφορά τους όμως προδίδει ημιμάθεια συνοδευόμενη από ένα κραυγαλέο κόμπλεξ κατωτερότητας. Έχουν οι καημένοι μια βαθιά ανάγκη να το παίζουν ξερόλες, δεν χάνουν ευκαιρία να αποπαίρνουν τους νεοφερμένους στο χώρο ή να επιτίθενται σε οποιοδήποτε εκλαμβάνουν ως επικίνδυνο για το μικρόκοσμό τους, δυσκολεύονται να προσπεράσουν ευκαιρία για τρολάρισμα και, το μόνο που δεν κάνουν, είναι ν' ανεβαίνουν σ' ένα βάθρο και να φωνάζουν "δοξάστε με!".

Αγνοήστε συστηματικά τέτοια τυπάκια, αν μη τι άλλο δεν αξίζουν το χρόνο σας. Ούτε για μια στιγμή μη μασήσετε, αν επιχειρήσουν να σας μειώσουν. Έχετε πάντοτε εμπιστοσύνη στις ικανότητές σας και επιδεικνύετε υπομονή κι επιμονή. Να είστε βέβαιοι ότι, πολύ πιο γρήγορα απ' όσο φαντάζεστε, όλοι αυτοί οι ξερόλες θα τρώνε τη σκόνη σας. Κι όταν αρχίζουν να χάνουν την ψυχραιμία τους και να βγάζουν νευράκια, μία είναι η λύση που αξίζει να τους προτείνετε.

Ξυδάκι.



ουέλ, μέλ

Αγαπητοί μου,

Επιτέλους καλοκαίρι! Μετά από δύο μήνες εκλογικής και μετεκλογικής εγρήγορσης, και μερικές βροχερές βδομάδες στην Γηραιά Αλβιώνα, έχω πλέον επιστρέψει στη βάση μου, στο Πόρτο Γερμενό. Η γραφομηχανή είναι ακουμπισμένη στον ξύλινο πάγκο δίπλα από τη μεγάλη αιώρα στην αυλή και εγώ είμαι καθισμένος μπροστά της και πατάω πλήκτρα.

Μοιάζει σαν να μην έχει περάσει ούτε μια μέρα από πέρσι, που καθισμένος και πάλι στην ίδια ακριβώς θέση ξεκίνησα να σας γράφω τις μηνιαίες μου επιστολές.

Και όμως έχουν περάσει έντεκα μήνες, και αυτή τη στιγμή ξεκινώ την δωδέκατη επιστολή μου.

Δεν σκοπεύω να κάνω ανασκόπηση της χρονιάς. Αυτό θα το κρατήσω για την επόμενη φορά. Αλλά κάποιες μελαγχολικές σκέψεις δεν μπορώ να τις αποφύγω.

Μελαγχολικές όχι τόσο για το γεγονός πως ο χρόνος περνάει -"πετάει", θα έγραφε ο ποιητής- όσο για το βάρος της μνήμης.

Τέλος πάντων, ας μην σας χαλάω το κέφι τώρα, χρονιάρες μέρες, οπότε let's stick to the facts: Το καλοκαίρι είναι εδώ, όλα είναι πολύ ήρεμα και ειδυλλιακά, και όλοι κοιμούνται για μεσημέρι, με την εξαίρεση των τζιτζικιών που (όπως και πέρσι) κοντεύουν να πάθουν αποπληξία από το πολύ τζι τζι.

Εχει πλάκα αυτό με τα τζιτζίκια. Για μας που είμαστε μόνιμα στην Ελλάδα, ο ήχος τους δεν μας κάνει και μεγάλη εντύπωση. Τζιτζίκια και καλοκαίρι πάνε μαζί. Για όλους τους εκτός, ωστόσο, αυτούς που επισκέπτονται την Ελλάδα όταν και όποτε, τα τζιτζίκια είναι κάτι το απρόσμενο και παράξενο. Για όλους τους Ελληνες, τέλος, που μένουν εκτός Ελλάδας, τα τζιτζίκια είναι το πρώτο πράγμα που νοσταλγούν όταν φτάνει το καλοκαίρι. Αυτό, τουλάχιστον, μου εκμυστηρεύεται ο ανιψιός.

Εκεί στα Λονδίνα, μου λέει, ειδικά όταν ο καιρός είναι καλός (η έμφαση εδώ πάει στο "όταν"), αυτό είναι που του λείπει περισσότερο, ο ήχος των τζιτζικιών.

Τζιτζίκια, λοιπόν.

Στο Πόρτο Γερμενό δεν έχουμε ίντερνετ, όπως ίσως θυμάστε, και έτσι έχω χάσει την άμεση επαφή μου με τον έξω κόσμο. Η Μαιρούλα έχει το αιΦόν της, βέβαια, και κάθε βδομάδα που ανεβαίνει στα Βίλια για ψώνια ενημερώνεται για την επικαιρότητα, και μας ενημερώνει κι εμάς επιστρέφοντας.

Πήγαν και σήμερα το πρωί με την Κυρία Καίτη, και γύρισαν γεμάτες νέα.

Έτσι το έχω μάθει, ας πούμε, ότι ο Ισημερινός έδωσε επί τέλους άσυλο στον Ασάνζ ή ότι στην Ύδρα οι εστιάτορες κήρυξαν νέο Ανένδοτο κατά του ΣΔΟΕ και της Αστυνομίας. Επίσης έχω μάθει ότι στην Χίο καίγονται και τα τελευταία δάση που τους έχουν απομείνει –είδα, μάλιστα, και σχετική φωτογραφία με κάτι χωρικές σε μια παραλία, όρθιες δίπλα από τις ξαπλώστρες και τους λοιπούς λουόμενους, να κοιτάζουν την φωτιά να πλησιάζει από το δάσος πιο κάτω. Έμαθα και για την συνεχιζόμενα πτωτική πορεία της μετοχής του Ζούπερμαν και φυσικά άκουσα τα πάντα για τον Ξένιο Δία, την εμπνευσμένη επιχείρηση της αστυνομίας με το εύστοχο όνομα.

Για τον Ασάνζ τι να πω; Με βάζει σε σκέψεις, βέβαια, το γεγονός ότι του δίνει άσυλο για μιαν υπόθεση σεξουαλικής παρενόχλησης η πρεσβεία μιας χώρας με κυβέρνηση που καταδικάζει δημοσιογράφους για τα κείμενά τους και κλείνει εφημερίδες επειδή έκαναν αντιπολιτευτικά σχόλια. Κάτι μου λέει ότι η «νίκη» του δεν είναι απλώς νίκη της ελευθεροτυπίας και της ελευθερίας στην πληροφόρηση. Από την άλλη, σε σκέψεις με βάζει και το γεγονός ότι οι Σουηδοί θυμήθηκαν την υπόθεση της παρενόχλησης μόλις αυτός ξεκίνησε τις αποκαλύψεις του. Οπότε τι;

Δεν έχω ιδέα.

Ούτε για την Ύδρα έχω να πω πολλά. Ξέρετε, τους καταλαβαίνω. Έχετε πάει στην Ύδρα; Οι άνθρωποι πεινάνε. Στα κοσμηματοπωλεία και τις μπουτίκ σκοτώνουν μύγες για να περάσει η ώρα. Η κίνηση από Ιαπωνία και Σαουδική Αραβία είναι πεσμένη φέτος, και από τα κρουαζιερόπλοια που προσδένουν στο λιμάνι, οι τουρίστες που κατεβαίνουν αρκούνται να φωτογραφίζουν τις γάτες και αποφεύγουν πλέον να κάτσουν με φόντο το ηλιοβασίλεμα για να παραγγείλουν μια ντοματοσαλάτα των δώδεκα ευρώ και έναν απλό εσπρέσο των πέντε.

Χώρια που φέτος στο νησί δεν έχει πατήσει ούτε ένας Γερμανός, και οι ντοματοσαλάτες έχουν μείνει απούλητες.

Προφανώς υπακούουν στις εντολές της Μέρκελ. Εμ, Γερμανοί είναι, τι περιμέναμε. Δεν τους ξέραμε δα; Πειθήνιοι. Μαλθακοί. Ταπεινωμένοι. Όχι σαν τους ηρωικούς τους Υδραίους που όπως τότε, στα ηρωικά



χρόνια του εικοσιένα, έτσι και σήμερα, μύγα στο σπαθί τους δεν ανέχονται και χαράτσια στον Σουλτάνο δεν πληρώνουν. Τίποτα δεν πληρώνουν. Ακόμα και διόδια να υπήρχαν στην Ύδρα, ούτε αυτά θα τα πλήρωναν.

Οπότε σιγά μην τον πληρώσουν τον φπα στα δώδεκα ευρώ της ντοματοσαλάτας. Ή στα πέντε ευρώ του εσπρέσο.

Ας τον πληρώσουν οι άλλοι. Ας τον πληρώσουν οι Γερμανοί. Αυτοί δεν παράγγειλαν την ντοματοσαλάτα; Αυτοί να πληρώσουν. Δηλαδή τι, μαζί την φάγαμε κι αυτή; Νισάφι!

Όπως και στη Χίο. Εμείς θα σβήσουμε τις φωτιές; Ένα μήνα τον έχουμε να δουλέψουμε για να βγάλουμε όλη τη χρονιά, θα αφήνουμε τα μαγαζιά για να τρέχουμε στα κατσάβραχα με τα χατζάρια και τις μάνικες; Και στο κάτω κάτω, τι κάνει το κράτος; Γιατί τους έχουμε εκεί; Για διακοσμητικούς; Να πάρει ο Πολύδωρας την κόρη του να πάνε αυτοί να σβήσουν τη φωτιά. Εμείς έχουμε και δουλειές.

Για τον Ζούπερμαν, τέλος, τι άλλο να πω; Η μετοχή πέφτει, οκ. Θα πέσει κι άλλο, οκ. Μεσ' στην θερινή ραστώνη και απομόνωση του Πόρτο Γερμενού, η μετοχή του Ζούπερμαν είναι χαμηλά στην ιεραρχία των πραγμάτων που με απασχολούν. Πολύ χαμηλά.

Όσο χαμηλά είναι και κάθε τι άλλο. Οφείλω να το παραδεχτώ. Στην πραγματικότητα τίποτα δεν με απασχολεί. Έχω παραδώσει τα όπλα.

Σας έγραφα τις προάλλες για τα Χρυσά Αβγά. Ίδρωσε το αφτί κανενός; Κανενός. Προβληματίστηκε κανείς για τα μαχαιρώματα που έχουν γίνει καθημερινή υπόθεση; Κανείς. Πιάσανε κανέναν για τα μαχαιρώματα αυτά; Κανέναν. Το μόνο που πιάνουνε, δηλαδή τι πιάνουνε, τους μαζεύουνε με το παραγάδι, είναι κάθε τι μελαμψό που κινείται στην ευρύτερη περιοχή πρωτευούσης και άλλων πόλεων.

Που λέτε, μεγάλη επιτυχία ο Ξένιος Ζευς. Από τους 10502 που προσήχθησαν, με ενημερώνει η Κυρία Καίτη, οι 1941, λέει, δεν είχανε χαρτιά και συνελήφθηκαν. Δηλαδή παρανομία σε ποσοστό 18,5%. Ή, για να το θέσω αλλιώς, νομιμότητα σε ποσοστό 81,5%.

Με ρώταγε αν αυτό είναι επαρκές ποσοστό για να δικαιολογήσει την έκταση (και το ύφος) της επιχείρησης; Ξέρω γω; Πείτε μου εσείς.

Για να το θέσω και με πιο abstract τρόπο, δικαιολογείται η οργάνωση εστιασμένων επιχειρήσεων «σκούπας» εναντίον τμημάτων ενός πληθυσμού όταν η παραβατικότητα δεν φαίνεται να ξεπερνά το 20%;

Εγώ δεν ξέρω τι να σκεφτώ.

Γιατί αν σκεφτώ ότι ναι, είναι δικαιολογημένη, τότε θα πρέπει να δηλώσω εθελοντής και να πάω στη Λευκάδα για να μαζέψω με το παραγάδι κάθε τι που κινείται και παρέχει τουριστικές υπηρεσίες. Κάθε τι. Από εστιάτορα μέχρι γάτα. Και μετά να πάω στην Ύδρα.

Διότι, όπως με πληροφορεί η μητέρα της Εξαδέλφης που είναι Ινδουίστρια, σύμφωνα με το ΣΔΟΕ στη Λευκάδα η σχετική παραβατικότητα ανέρχεται στο 100%.

Να, τώρα, όπου να' ναι ξυπνάει από τον μεσημεριανό υπνάκο, και θα έρθει να σας το πει η ίδια.

Και βγάλτε σείς τα συμπεράσματά σας. Εγώ σας το είπα από την αρχή. Δεν έχω ιδέα. Και ούτε που με ενδιαφέρει.

Αν δεν σας ενδιαφέρει εσάς.

Σας ασπάζομαι,
Θείος Ακάκιος



Πλήρη **άρθρα που συμπληρώνουν** το περιοδικό. Τα **εβδομαδιαία** μας **video casts** [έρχονται, έρχονται]. Ολόκληρα **blasts from the past**. Τα **βίντεο** από τα **live streams**. Τα πάντα ενδιαφέροντα **forums**. Το **σημείο συνάντησης** για τους φίλους του περιοδικού. Η πιο **ζωντανή κοινότητα** και οι πιο **ανήσυχοι, δημιουργικοί χρήστες**!

deltahacker.gr

What have you hacked for me lately?

Skill: Beginner

Tags: malware, registry, auto start, process explorer, task manager

Ξετρυπώστε τα ζιζάνια!

by Thiseas



Η μόλυνση ενός συστήματος Windows είναι τόσο συχνό φαινόμενο, που έχουμε καταντήσει να το θεωρούμε φυσιολογικό. Αυτή τη φορά δεν θα αναζητήσουμε τα αίτια που οδηγούν σ' αυτή την κατάσταση, αλλά θα μάθουμε να την αντιμετωπίζουμε: Θα δούμε πώς εντοπίζονται και πώς απομακρύνονται τα κακόβουλα προγράμματα... χειροκίνητα! Γιατί πώς να το κάνουμε, υπάρχουν περιπτώσεις που το anti-virus πιάνεται στον ύπνο...

Δεν σκοπεύουμε να σας κουράσουμε με μια ανούσια φιλολογική συζήτηση γύρω από το malware. Δεν πρόκειται να περιγράψουμε τα διάφορα είδη του, ούτε θα επιχειρήσουμε να δώσουμε όνομα σε κάθε είδος (όπως προσπαθούν να κάνουν πολλοί). Αυτό που μας ενδιαφέρει είναι να γνωρίσουμε και να εξετάσουμε τα ζωτικά σημεία ενός συστήματος Windows, στα οποία επιτίθενται τα περισσότερα malware (viruses, worms κι οτιδήποτε άλλο περνάει από το νου σας). Επιπρόσθετα, θα αναφέρουμε και μερικούς τρόπους με τους οποίους μπορούμε ν' ανακαλύπτουμε malware που έχει χωθεί στο σύστημά μας, καθώς και το πώς μπορούμε να το απομακρύνουμε. Όλα αυτά θα τα κάνουμε χειροκίνητα, χωρίς την χρήση "αντιβιοτικών" ή "καθαριστικών" προγραμμάτων. Όχι ότι σας προτείνουμε να μην έχετε antivirus ή κάτι παρόμοιο, αλλά σε κάποιες περιπτώσεις τα προγράμματα του είδους αποδεικνύονται ανεπαρκή, οπότε οφείλουμε να γνωρίζουμε πώς ν' αναλαμβάνουμε δράση. Στην παρουσιασή μας θα θεωρήσουμε ότι το σύστημα που δέχεται επίθεση είναι ένα box με Windows 7. Αυτό βέβαια δεν πρέπει να αγχώσει τους παραδοσιακούς χρήστες. Όσα θα δούμε ισχύουν και στην περίπτωση των Windows XP και των Vista. Πριν ξεκινήσουμε, να ξεκαθαρίσουμε ότι δεν πρόκειται να παρουσιάσουμε κάποια μαγική συνταγή. Δεν πρόκειται να δώσουμε ένα φάρμακο για την αντιμετώπιση όλων των malware. Το μόνο σίγουρο όμως είναι ότι θα παρουσιάσουμε όλα όσα πρέπει να γνωρίζει κανείς, για να ξεκινήσει τον έλεγχο και τον καθαρισμό του συστήματός του.

Τρωτά σημεία

Ποια μέρη του συστήματος προτιμούν τα περισσότερα malware; Για να απαντήσουμε σ' αυτό το ερώτημα, πρέπει πρώτα να μελετήσουμε τη συμπεριφορά που παρουσιάζουν αυτά τα προγράμματα. Σ' αυτή τη μελέτη θα επικεντρώσουμε την προσοχή μας σε δύο βασικά στοιχεία της συμπεριφοράς του malware:

- Στον μηχανισμό με τον οποίο εκτελούνται *αφού* εγκατασταθούν στο σύστημα.
- Στην περιοχή του δίσκου, στην οποία επιλέγουν να εγκατασταθούν.

Να τονίσουμε ότι, τουλάχιστον σ' αυτό το άρθρο, δεν θα ασχοληθούμε καθόλου με τον τρόπο διείσδυσης του malware σ' ένα σύστημα. Τις περισσότερες φορές πρόκειται για κάποια αδυναμία zero-day του λειτουργικού συστήματος ή κάποιου άλλου προγράμματος (π.χ., του Firefox, του IE, κ.ο.κ.). Εμείς θα ασχοληθούμε με όσα συμβαίνουν *μετά* από μια επιτυχημένη εισβολή. Αυτή τη φορά, το θέμα μας είναι να καταπολεμήσουμε το εγκατεστημένο malware κι όχι να κλείσουμε τις τρύπες του συστήματος ή άλλων εφαρμογών.

Μηχανισμός εκκίνησης

Πρέπει να ξέρετε ότι το malware έχει αρκετά κοινά χαρακτηριστικά με τα "απλά" προγράμματα. Αυτά τα ηλεκτρονικά ζιζάνια είναι προγράμματα σαν όλα τ' άλλα, όπως το Word, το Excel και πάει λέγοντας. Για να επιτελέσουν το κακόβουλο έργο τους, πρέπει κάποιος να τα ενεργοποιήσει. Μ' άλλα λόγια, κάποιος πρέπει να τα τρέξει. Κάπου εδώ εντοπίζεται ένα βασικό τρικ που πραγματοποιούν τα προγράμματα του είδους: Τρέχουν μόνα τους! Χμ, εντάξει, σχεδόν μόνα τους. Όπως σ' όλες τις επιστήμες, έτσι και στην πληροφορική τίποτα δεν γίνεται με μαγικό τρόπο. Κανένα πρόγραμμα δεν μπορεί να τρέξει «από μόνο του». Στην πραγματικότητα,

το malware εκτελείται συνήθως από κάποιον μηχανισμό αυτόματης εκκίνησης, τον οποίο φέρει το λειτουργικό σύστημα (Σ.τ.Ε. Άλλο πρόγραμμα αυτό. Ε, τι, να μην προσθέσω *κι εδώ* κάτι; :D) Μάλιστα, τα περισσότερα προγράμματα του είδους προτιμούν να εκτελούνται ακριβώς μετά την εκκίνηση του λειτουργικού. Αυτό μπορεί να γίνει με διάφορους τρόπους κι ο συνηθέστερος σχετίζεται με το registry των Windows. Συγκεκριμένα, στο registry υπάρχουν μερικά κλειδιά, των οποίων οι καταχωρήσεις εκτελούνται αυτόματα. Για παράδειγμα, δείτε τα ακόλουθα:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
```

Οτιδήποτε καταχωρηθεί "κάτω" από αυτά τα κλειδιά, αντιμετωπίζεται σαν νόμιμο πρόγραμμα και καλείται αυτόματα, μετά από κάθε εκκίνηση του λειτουργικού συστήματος. Τα παραπάνω κλειδιά αφορούν σε όλους τους χρήστες του λειτουργικού, αφού βρίσκονται κάτω από το LocalMachine. Δώστε προσοχή στο κλειδί RUN. Αντίστοιχα κλειδιά υπάρχουν για κάθε χρήστη ξεχωριστά, κάτω από το HKEY_CURRENT_USER, αλλά και για τον λεγόμενο default user, κάτω από το HKEY_USERS\.DEFAULT\. Εκτός από το registry, τα Windows περιλαμβάνουν κι άλλους μηχανισμούς για την αυτόματη εκκίνηση προγραμμάτων. Το σύστημα ελέγχει κι ορισμένες τοποθεσίες στο δίσκο, όπως οι ακόλουθες:

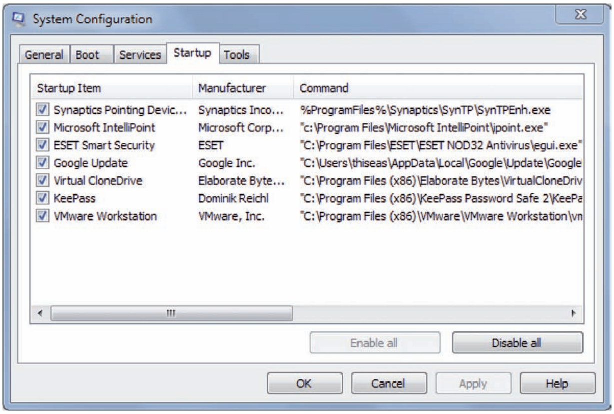
```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup
C:\Users\<USER-NAME>\AppData\Roaming\Microsoft\Windows\Start
Menu\Programs\Startup
```

Εκεί είναι δυνατό να βρίσκονται διάφορες συντομεύσεις προς προγράμματα, τα οποία εκτελούνται αυτόματα κατά το λεγόμενο startup των Windows. Όπως μπορείτε να καταλάβετε, το registry του συστήματος και οι κατάλογοι που αναφέραμε παραπάνω αποτελούν κλασικές τοποθεσίες στις οποίες τρυπώνουν τα κακόβουλα προγράμματα. Σ' αυτό το σημείο, για λόγους πληρότητας πρέπει να αναφέρουμε τα αρχεία C:\Windows\Win.ini και C:\Autoexec.bat. Αυτά διαβάζονται κατά την εκκίνηση του λειτουργικού και, μεταξύ άλλων, μπορούν να εκτελούν άλλα προγράμματα. Επομένως, ένα πρόγραμμα malware θα μπορούσε να προσβάλλει τα εν λόγω αρχεία, ώστε να εξασφαλίσει την αυτόματη εκκίνησή του.

Πρώτη αντίδραση

Αν υποπτευόμαστε ότι το σύστημά μας έχει μολυνθεί από malware, τότε οφείλουμε να ξεκινήσουμε τσεκάροντας τα προγράμματα που εκτελούνται αυτόματα. Για το σκοπό αυτό μπορούμε να τρέξουμε το εργαλείο msconfig[1] και να μεταβούμε στην καρτέλα Startup (βλ. εικόνα 1). Εκεί θα αναζητήσουμε παράξενες εγγραφές. Δηλαδή, προγράμματα που έχουν εντελώς άγνωστο όνομα, παράξενο ή απλά ύποπτο! Κατά πάσα πιθανότητα, τα προγράμματα που έχουν αυτά τα χαρακτηριστικά αποτελούν

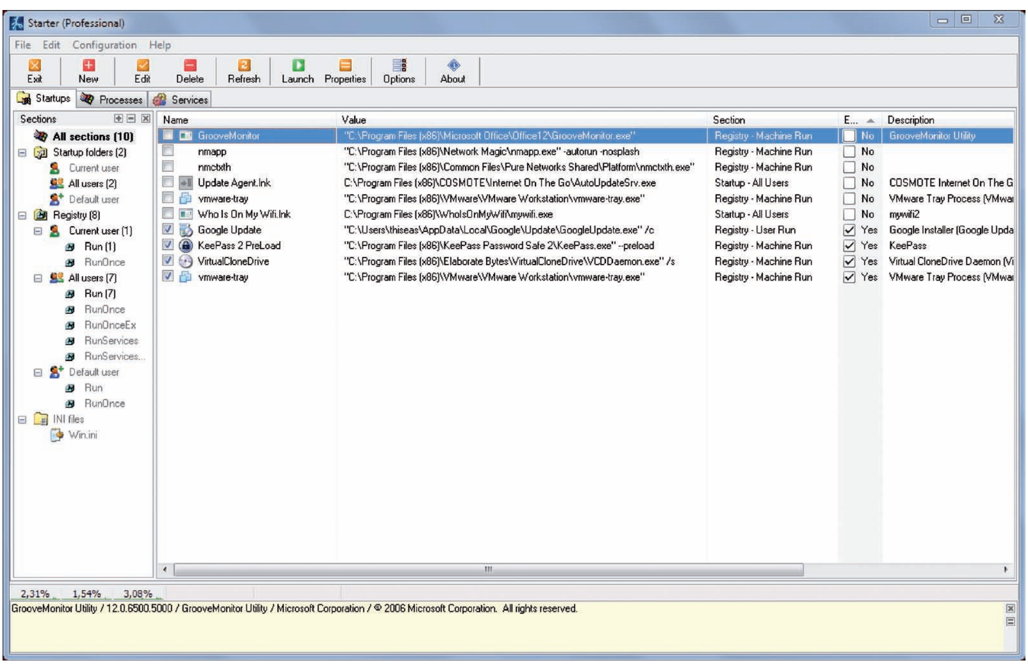
Εικόνα 1
Από την καρτέλα startup του εργαλείου msconfig μπορούμε να δούμε ποια προγράμματα ξεκινούν αυτόματα, κατά την εκκίνηση του λειτουργικού συστήματος.



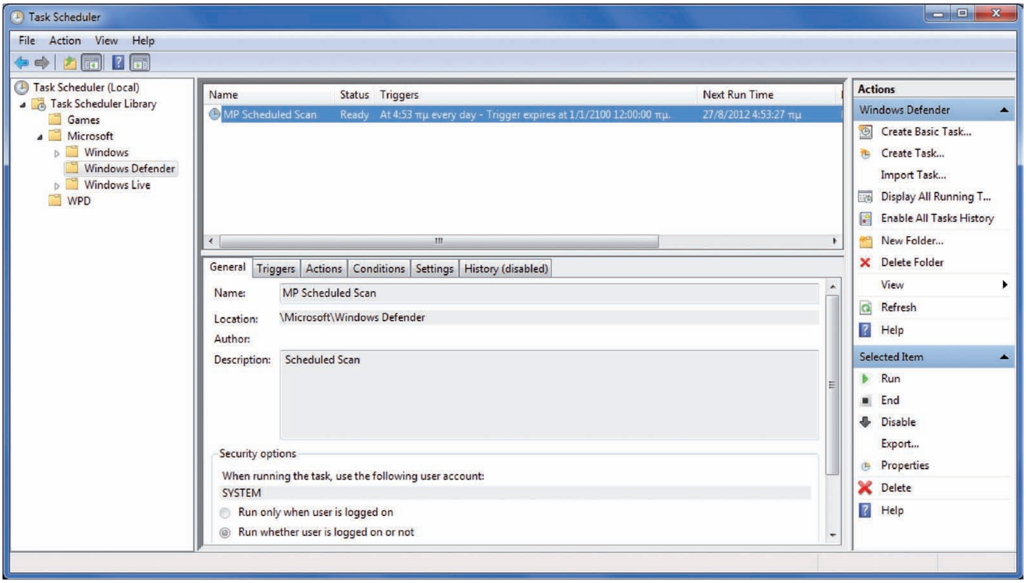
1. Στο ίδιο εργαλείο καταλήγουμε και με την εξής διαδρομή: Start → Control Panel → Administrative Tools → System Configuration.

malware. Ας το εξηγήσουμε λίγο αυτό: Προφανώς, κάθε πρόγραμμα πρέπει να έχει ένα όνομα. Οι προγραμματιστές του malware, λοιπόν, δίνουν τυχαία ονόματα στα προγράμματα τους, τα οποία δημιουργούνται αυτόματα και δεν είναι πάντοτε τα ίδια. Ο λόγος για τον οποίο γίνεται αυτό είναι η απόκρυψη. Αν κάποιο malware είχε πάντοτε το ίδιο όνομα, θα ήταν πολύ εύκολο να εντοπιστεί από ένα πρόγραμμα καθαρισμού, όπως επίσης κι από τους ίδιους τους χρήστες. Έτσι, πολλοί δημιουργοί χρησιμοποιούν ονόματα που αποτελούνται από μια σειρά τυχαίων χαρακτήρων (π.χ., sdfhgjd.exe). Κάποιοι άλλοι ακολουθούν εντελώς διαφορετική τακτική: Επιλέγουν ονόματα υπεράνω πάσης υποψίας, όπως, π.χ., msword.exe, systemio.exe κ.ο.κ. Σ' αυτές τις περιπτώσεις χρειάζεται περισσότερη προσοχή. Για παράδειγμα, αν βρούμε στην αυτόματη εκκίνηση την εγγραφή C:\Windows\msword.exe, το πιο πιθανότερο είναι να μην πρόκειται για το γνήσιο MS Word αφού αυτό δεν εγκαθίσταται στον βασικό κατάλογο του συστήματος, αλλά σε μία πολύ συγκεκριμένη θέση μέσα στο Program Files. Από την άλλη, αν κι έχει ψαρωτικό όνομα, το systemio.exe είναι εντελώς άγνωστο! Βέβαια ένας έμπειρος χρήστης γνωρίζει ότι δεν τρέχει κανένα πρόγραμμα με αυτό το όνομα -- όχι σ' ένα υγιές σύστημα Windows. Όπως καταλαβαίνετε, λοιπόν, στην αναζήτηση κακόβουλων προγραμμάτων ισχύει το εξής: Όσο πιο καλά γνωρίζει κανείς τα προγράμματα που εκτελούνται υπό φυσιολογικές συνθήκες, τόσο πιο πολλές είναι οι πιθανότητες να ξεθάψει κάποιο κρυμμένο malware. Κάπως έτσι προκύπτει ο ακόλουθος χρυσός κανόνας: **Καλό είναι να γνωρίζετε τα (βασικά) εξαρτήματα του συστήματος και τα προγράμματα που έχετε εγκαταστήσει εσείς, όπως επίσης το πώς ονομάζονται και το πού βρίσκονται.**

Για να διευκολύνετε τη ζωή σας, μπορείτε να χρησιμοποιείτε κάποιο από τα freeware εργαλεία που διαχειρίζονται τις εγγραφές στην αυτόματη εκκίνηση του συστήματος. Ο γράφων χρησιμοποιεί (και προτείνει) το



Εικόνα 2
Το δωρεάν εργαλείο Starter προβάλει όλα τα προγράμματα που ξεκινούν αυτόματα, άσχετα αν οι σχετικές αναφορές έχουν προστεθεί στο registry ή σε κάποιον κατάλογο αυτόματης εκκίνησης. Από το ίδιο το Starter μπορούμε να απενεργοποιήσουμε την αυτόματη εκκίνηση ενός προγράμματος και, εφόσον το επιθυμούμε, να διαγράψουμε πλήρως και τη σχετική αναφορά στο registry του συστήματος.



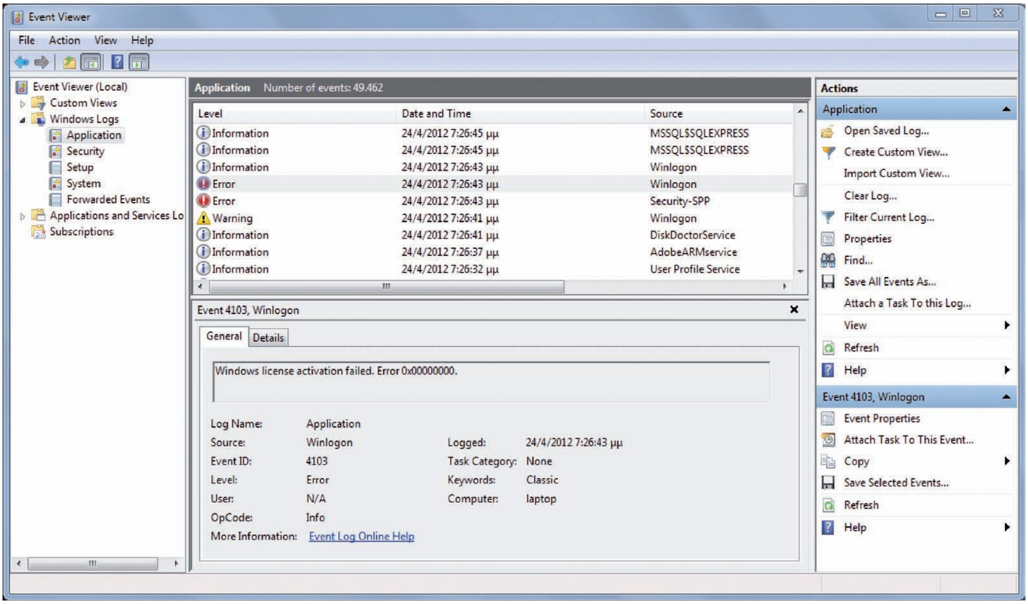
Εικόνα 3
Όταν ψάχνουμε για malware, πρέπει να τσεκάρουμε οτιδήποτε εκτελείται εκείνη την ώρα στον υπολογιστή, καθώς κι οτιδήποτε ενδέχεται να εκτελεστεί στο μέλλον. Ακριβώς γι' αυτό, μια βόλτα από τον Task Scheduler κρίνεται απαραίτητη.

Starter, από την CodeStaf (www.softpedia.com/get/Tweak/System-Tweak/Starter.shtml).

Αντιμετώπιση

Ας υποθέσουμε ότι ο υπολογιστής μας παρουσιάζει παράξενη συμπεριφορά κι ότι μόλις ανακαλύψαμε μια ύποπτη εγγραφή, στο registry ή στους καταλόγους αυτόματης εκκίνησης. Σ' αυτή την περίπτωση πρέπει να ακολουθήσουμε τα εξής βήματα:

1. Κάνουμε επανεκκίνηση του συστήματος σε Safe Mode (http://en.wikipedia.org/wiki/Safe_mode). Για το σκοπό αυτό πατάμε το [F8], αμέσως μετά το POST του BIOS και όταν ξεκινά το boot των Windows. Προφανώς, στο μενού που θα εμφανιστεί πρέπει να επιλέξουμε το "Safe Mode".
2. Ανοίγουμε το πρόγραμμα msconfig (ή το Starter ή όποιο άλλο χρησιμοποιούμε) και εντοπίζουμε τις ύποπτες εγγραφές.
3. Σημειώνουμε τις θέσεις (τα paths) στις οποίες βρίσκονται τα πιθανώς επικίνδυνα προγράμματα.
4. Απενεργοποιούμε τη σχετική εγγραφή, χωρίς να τη διαγράψουμε εντελώς. Αυτό γιατί ενδέχεται να έχουμε κάνει λάθος και να πρόκειται για κάτι χρήσιμο.
5. Ανοίγουμε τον Windows Explorer, μεταβαίνουμε στις θέσεις που σημειώσαμε νωρίτερα και μετακινούμε τα ύποπτα αρχεία σε κάποια άλλη, ασφαλή θέση. Για παράδειγμα, σε κάποιον κατάλογο που φτιάξαμε επίτηδες, σε κάποιον εξωτερικό δίσκο.
6. Κάνουμε reboot (σε "Normal Mode") κι ελέγχουμε τη συμπεριφορά του συστήματος.
7. Αν λειτουργούν όλα φυσιολογικά, μπορούμε να διαγράψουμε οριστικά τα αρχεία που είχαμε μετακινήσει προηγουμένως, όπως και



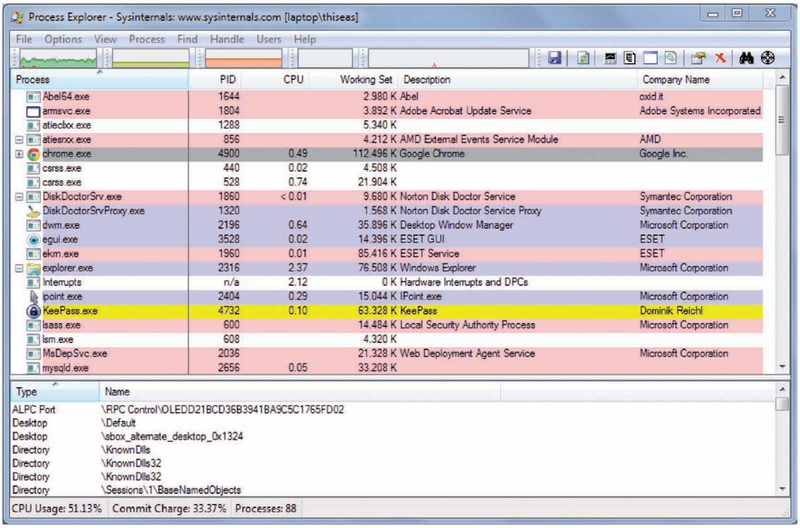
Εικόνα 4
Ο Event Viewer προσφέρει πρόσβαση σε όλα τα "κρυμμένα" μηνύματα του συστήματος. Ο έλεγχος αυτών των μηνυμάτων είναι κουραστικός (και λίγο βαρετός), αλλά μέσα από αυτή τη διαδικασία ενδέχεται να εντοπίσουμε τα ίχνη κάποιου κακόβουλου προγράμματος. Εν ολίγοις, ο Event Viewer αποτελεί απαραίτητο εργαλείο ανασκαφής!

τις σχετικές εγγραφές στο registry και στους καταλόγους αυτόματης εκκίνησης.

8. Κάνουμε μια ακόμα επανεκκίνηση και ελέγχουμε πάλι για ύποπτα προγράμματα που ενδέχεται να ξεκινούν αυτόματα.

Περιοχή εγκατάστασης

Υπάρχουν μερικές τοποθεσίες που θα λέγαμε ότι αποτελούν δημοφιλείς προορισμούς για το malware. Φυσικά, ένα κακόβουλο πρόγραμμα θα μπορούσε να αποφύγει τις καθιερωμένες θέσεις και να τρυπώσει οπουδήποτε! Ωστόσο είναι χρήσιμο να γνωρίζουμε τις εν λόγω θέσεις, ώστε να ξέρουμε από πού να ξεκινάμε την αναζήτηση κακόβουλου λογισμικού και, γενικότερα, να έχουμε έναν συστηματικό τρόπο αναζήτησης. Το malware, λοιπόν, τρυπώνει συνήθως στον κατάλογο όπου βρίσκονται τα αρχεία του λειτουργικού συστήματος (C:\Windows). Μια άλλη συνηθισμένη θέση είναι ο κατάλογος C:\temp, ο οποίος χρησιμοποιείται από το σύστημα ως προσωρινός αποθηκευτικός χώρος. Βέβαια στα Windows Vista και στις μεταγενέστερες εκδόσεις, η εγγραφή σε αυτόν τον κατάλογο απαγορεύεται χωρίς τη συγκατάθεση του χρήστη (βλέπε UAC, `http://en.wikipedia.org/wiki/User_Account_Control`). Ακριβώς γι' αυτό, το malware καταφεύγει συχνά στον κατάλογο C:\Users\<USER-NAME>\AppData\Local\Temp. Αυτός ο κατάλογος είναι αντίστοιχος του C:\temp, αλλά ανήκει εξ ολοκλήρου στον χρήστη USER-NAME και η εγγραφή εκεί μέσα επιτρέπεται ελεύθερα, χωρίς την ενεργοποίηση του UAC. Τέλος, αξίζει να αναφέρουμε και την περίπτωση ορισμένων εξελιγμένων ζιζανίων, τα οποία φροντίζουν να αλλάζουν θέση ή/και όνομα σε τακτά χρονικά διαστήματα. Αυτό το πετυχαίνουν με τη βοήθεια του Task Scheduler των Windows. Για όσους δεν γνωρίζουν, ο Task Scheduler είναι ένα εργαλείο που συνοδεύει τα Windows και το οποίο επιτρέπει την προγραμματισμένη εκτέλεση προγραμμάτων, είτε περιοδικά είτε σε κάποια δεδομένη χρονική στιγμή. Για παράδειγμα, η αυτόματη λήψη backup των αρχείων συστήματος αποτελεί μια εργασία που έχει ρυθμιστεί μέσα από τον Task Scheduler.



Εικόνα 5
Από τον Process Explorer δεν κρύβεται (σχεδόν) τίποτα. Με τη βοήθειά του μπορούμε να τσεκάρουμε εύκολα αν εκτελείται κάποιο ύποπτο πρόγραμμα και, φυσικά, να το τερματίσουμε άμεσα!

Με λίγα λόγια, το εν λόγω πρόγραμμα αποτελεί το ισοδύναμο του Cron Daemon, που διαθέτουν τα συστήματα Unix. Μπορούμε να βρούμε τον Task Scheduler και να ελέγξουμε τις προγραμματισμένες εργασίες του, ακολουθώντας της εξής διαδρομή: *Start → Control Panel → All Control Panel Items → Administrative Tools → Task Scheduler* (βλ. εικόνα 3).

Ύποπτες ενδείξεις

Υπάρχουν ορισμένα προγράμματα που εμφανίζουν τα κρυφά μηνύματα του λειτουργικού συστήματος. Αυτά τα προγράμματα, αν και δεν δημιουργήθηκαν για την αποκάλυψη του malware, αποτελούν ιδιαίτερα χρήσιμα εργαλεία σε αυτή τη δουλειά. Ένα τέτοιο εργαλείο είναι το περίφημο Event Viewer (http://en.wikipedia.org/wiki/Event_Viewer) και μπορούμε να το βρούμε ακολουθώντας την εξής διαδρομή: *Start → Control Panel → All Control Panel Items → Administrative Tools → Event Viewer* (βλ. εικόνα 4). Με τον Event Viewer μπορούμε να βλέπουμε άμεσα τα απλά, τα προειδοποιητικά μηνύματα, όπως και τις αναφορές σφαλμάτων από όλα τα προγράμματα και τις υπηρεσίες του συστήματος. Όπως φαίνεται και στην εικόνα 4, κάτω από το Windows Logs υπάρχουν οι κατηγορίες μηνυμάτων Application, Security και System. Ξεκινώντας λίγο ανάποδα, να πούμε ότι στο System *δεν* πρέπει να εμφανίζεται καμία αναφορά σφαλμάτων. Αν διαπιστώσουμε κάτι τέτοιο τότε οφείλουμε να το διερευνήσουμε άμεσα, διότι κάποιο πρόγραμμα (συνήθως κάποιος driver) ενδέχεται να μη συνεργάζεται καλά με το hardware. Επίσης, ένα μήνυμα λάθους στην εν λόγω κατηγορία μπορεί να σημαίνει πραγματική βλάβη στο hardware. Στην κατηγορία Security αναφέρονται όλες οι παρασκηνιακές κινήσεις που πραγματοποιούν τα Windows και οι οποίες σχετίζονται με την ασφάλεια. Εκεί για παράδειγμα θα δείτε τις συνδέσεις που πραγματοποιούνται στο σύστημα (τα login) κ.λπ. Προφανώς, θα πρέπει να κοιτάξουμε κι εκεί μέσα! Στις περισσότερες των περιπτώσεων δεν θα δούμε τίποτε περίεργο. Τέλος, στην κατηγορία Application πρέπει να ρίξουμε το μεγαλύτερο βάρος. Εδώ ενδέχεται να κρύβονται διάφορα ύποπτα μηνύματα, όπως η αναφορά για μία επίθεση buffer overrun. Κανονικά, πρέπει να δίνουμε ιδιαίτερη σημασία σε όλα τα Errors και τα Warnings. Πρόκειται για μια αρκετά κοπιαστική, χρονοβόρα και βαρετή διαδικασία, αλλά κά-

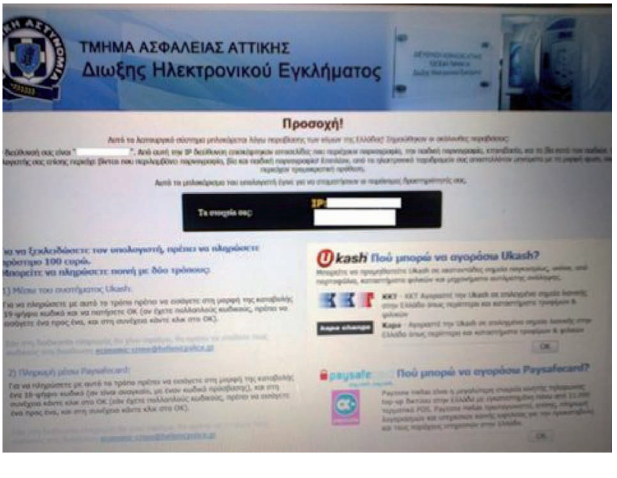
ποιες φορές είναι απαραίτητη κι ενδέχεται να ευχαριστούμε την καλή μας τύχη που αποφασίσαμε ν' ασχοληθούμε.

Εντοπισμός και τερματισμός

Ένα πολύτιμο εργαλείο στην αντιμετώπιση του malware είναι ο Task Manager. Από αυτόν μπορούμε να δούμε όλες τις εργασίες που εκτελούνται στο παρασκήνιο του συστήματος. Για να τον εμφανίσουμε αρκεί να πατήσουμε το συνδυασμό πλήκτρων [CTRL+Shift+ESC]. Βέβαια πρέπει να σημειώσουμε ότι υπάρχει ένα πολύ καλύτερο εργαλείο για αυτή δουλειά, το οποίο διανέμεται δωρεάν. Αναφερόμαστε στον Process Explorer (http://technet.microsoft.com/en-us/sysinternals/bb896653). Αρκεί να σας πούμε ότι το εν λόγω εργαλείο προτείνεται και από την ίδια τη μαμά Microsoft.

Αναρωτιέστε πώς εργαζόμαστε με αυτό το εργαλείο; Στο δίκτυο κυκλοφορούν πάρα πολλοί οδηγοί χρήσης και μια σχετική περιγραφή θα ξέφευγε από το θέμα μας. Θα σταθούμε μόνο σε μια δυνατότητα του προγράμματος, την οποία θεωρούμε ιδιαίτερα εξυπηρετική. Μιλάμε για τη δυνατότητα αναζήτησης ενός προγράμματος, ενός DLL ή μιας υπηρεσίας που τρέχει στο σύστημα. Ας υποθέσουμε, λοιπόν, ότι έχουμε εντοπίσει ένα ύποπτο πρόγραμμα στην αυτόματη εκκίνηση του συστήματος. Για να τσεκάρουμε αν εκτελείται ακόμα, αρκεί να τρέξουμε τον Process Explorer και να πατήσουμε [CTRL+F]. Έτσι, θα ανοίξει ένα παράθυρο στο οποίο μπορούμε να εισαγάγουμε το όνομα (ολόκληρο ή ένα τμήμα) του ύποπτου προγράμματος. Όταν πατήσουμε [ENTER], το πρόγραμμα θα πραγματοποιήσει τη σχετική αναζήτηση και θα μαρκάρει τα όποια αποτελέσματα. Αν εντοπίσουμε το ύποπτο πρόγραμμα, μπορούμε να τερματίσουμε το ίδιο και όλα τα παράγωγά του (διεργασίες που έχει ξεκινήσει το ίδιο), κάνοντας δεξί κλικ επάνω του και επιλέγοντας το Kill Process Tree. Όπως αντιλαμβάνεστε, με αυτή την ενέργεια διακόπτουμε την εκτέλεση του προγράμματος και το απομακρύνουμε από τη μνήμη του συστήματος. Ωστόσο, το ίδιο το πρόγραμμα παραμένει στο δίσκο μας και στην επόμενη εκκίνηση του θα ενεργοποιηθεί ξανά. Για να το απομακρύνουμε μόνιμα πρέπει να ακολουθήσουμε τη διαδικασία των οκτώ βημάτων, την οποία αναφέραμε παραπάνω.

Εικόνα 6
Το βλακώδες worm εν δράση ;)



Το σκουλήκι του νόμου

Σχετικά πρόσφατα έκανε την εμφάνιση του ένα worm, στο οποίο δόθηκε εσφαλμένα ο χαρακτηρισμός "ιός της Αστυνομίας". Πρόκειται για ένα πονηρό worm, το οποίο εγκαθίσταται στη θέση C:\Windows και καλείται αυτόματα από το registry με το ακόλουθο κλειδί:

```
HKEY_LOCAL_MACHINE\
SOFTWARE\Microsoft\Windows\
CurrentVersion\Run
```

Εφόσον το worm ενεργοποιηθεί, κάθε φορά που ξεκινάει το σύστημα εμφανίζεται το σήμα της δίωξης ηλεκτρονικού εγκλήματος μαζί με το ακόλουθο μήνυμα:

Προσοχή:

Αυτό το λειτουργικό σύστημα μπλοκάρεται λόγω παραβίασης των νόμων της Ελλάδας! Σημειώθηκαν οι ακόλουθες παραβάσεις:

Η IP διεύθυνσή σας είναι ..... Από αυτή την IP διεύθυνση επισκέφτηκαν ιστοσελίδες που περιέχουν πορνογραφία, παιδική πορνογραφία, κτηνοβασία και βία κατά των παιδιών. Ο υπολογιστής σας επίσης περιείχε βίντεο που περιλαμβάνει πορνογραφία, βία και παιδική πορνογραφία. Επιπλέον, από το ηλεκτρονικό σας ταχυδρομείο αποστέλλονταν μηνύματα με τη μορφή spam, που περιείχαν τρομοκρατική πρόθεση.

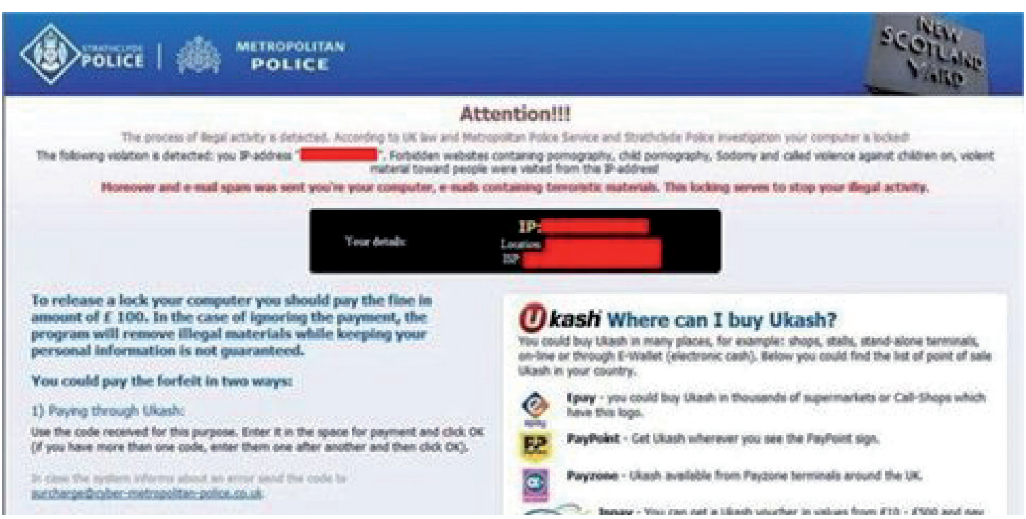
Αυτό το μπλοκάρισμα του υπολογιστή έγινε για να σταματήσουν οι παράνομες δραστηριότητές σας.

Για να ξεκλειδώσετε τον υπολογιστή, πρέπει να πληρώσετε πρόστιμο 100 ευρώ.

Μπορείτε να πληρώσετε την ποινή με δύο τρόπους...

«Λόγω παραβίασης των νόμων της Ελλάδας...», LOL! Πρόκειται για ένα εντελώς ανόητο μήνυμα, το οποίο ποντάρει στην άγνοια και στο φόβο ορισμένων χρηστών. Η απάτη αυτού του worm κάνει μπαμ από χιλιόμετρα. Κατ' αρχάς, τα ελληνικά του θυμίζουν μετάφραση από το Google. Κανένας έλληνας (και πολύ περισσότερο άνθρωπος της πληροφορικής) δεν θα έγραφε κείμενο με τόσα λάθη. Στο κάτω-κάτω, αν η δίωξη ηλεκτρονικού εγκλήματος εντόπιζε μια παρανομία, δεν θα προσπαθούσε να κλειδώσει τον υπολογιστή του εγκληματία, αλλά να τον συλλάβει. Αν κολλήσετε το συγκεκριμένο worm, λοιπόν, δεν πρέπει να ψαρώσετε ούτε στιγμή! Ακολουθήστε τα οκτώ βήματα που περιγράψαμε νωρίτερα και θα ξεμπερδέψετε οριστικά. Ο γράφων αντιμετώπισε το συγκεκριμένο worm και δεν χρειάστηκε να κάνει restore σε προηγούμενο restore point, όπως εσφαλμένα αναφέρουν διάφορα sites. Τέλος, κατά πάσα πιθανότητα το εν λόγω worm αποτελεί μετάφραση κάποιου ξένου. Για του λόγου το αληθές δείτε την εικόνα 7.

Εικόνα 7
Το ίδιο βλακώδες worm στη μητρική του γλώσσα, τα Αγγλικά!



Skill: Intermediate

Tags: wpad, proxy, spoofing, NetBIOS

WPAD Man-in-The-Middle Attack

by 41i3n

Μια φλόγα (όχι αναγκαία η Ολυμπιακή) αποδεικνύεται αρκετή να μας βάλει φωτιά μέσα στο καλοκαίρι. Πριν όμως κάνετε πράξη το μότο της Πυροσβεστικής Υπηρεσίας «Κάνε αμέσως κάτι» και καλέσετε το 199, αναλογιστείτε μήπως μπορείτε να αναβαθμίσετε το γεγονός σε πρώτης τάξης learning experience! Αν πάλι σας έχουμε μπερδέψει, μη σκάτε, ακόμα δεν αρχίσαμε! Δύο είναι τα πιθανά ενδεχόμενα στο δειγματοχώρο μας: Ή συμπέρασμα θα βγάλουμε ή στάχτη όλα θα γίνουν!

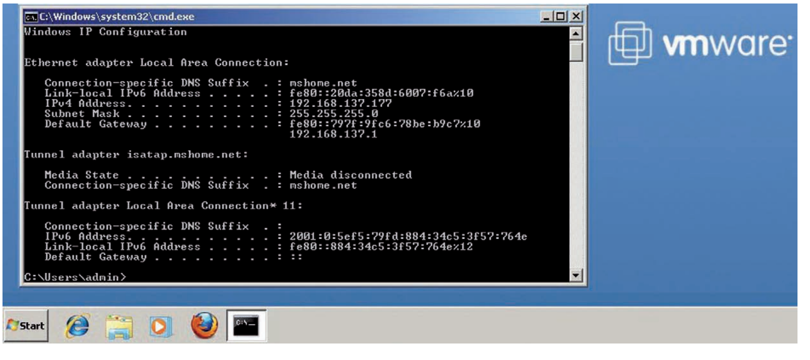
Όπως θα έπρεπε να γνωρίζετε, οι εξωγήινοι δεν έχουν ανάγκη να φορτίσουν τις μπαταρίες τους, δεν μετακινούνται από το Α στο Β αν δεν υπάρχει σοβαρός λόγος, ούτε έχουν κάποια άλλη ανθρώπινη ανάγκη. Έτσι, ένα editorial σαν εκείνο του deltahacker 010 τούς φαίνεται ακατανόητο -- αν και ομολογουμένως διασκεδαστικό (`http://deltahacker.gr/2012/07/11/deltahacker010editorial`). Οι εξωγήινοι ασχολούνται με κάθε λογής ασήμαντα πράγματα κι αναλώνονται σε λεπτομέρειες που κανένας άλλος δεν θα πρόσεχε. Έτσι, ο εξωγήινος της διπλανής πόρτας, δηλαδή ο γράφων, πέρασε αρκετό χρόνο το καλοκαίρι μελετώντας την αναπαραγωγή ενός σχετικά νέου malware, με το όνομα Flame (`http://en.wikipedia.org/wiki/Flame_(malware)`).

Το να μιλάω για τον εαυτό μου σε τρίτο πρόσωπο είναι τουλάχιστον παράξενο, γι’ αυτό και το γυρίζω αμέσως σε πρώτο.

Αυτό που τράβηξε την προσοχή μου, λοιπόν, ήταν ότι το Flame αναπαράγεται αξιοποιώντας μια ευπάθεια του 2007. Αναφέρομαι στο WPAD MiTM attack (`http://bit.ly/msupdmitm`). Μερικούς μήνες πριν, συζητούσαμε με τον subZraw για το πόσο έχουν εξελιχθεί τα νέα συστήματα και σχολιάζαμε ότι είναι σχεδόν αδύνατο να προσβληθούν από μια τόσο χαμηλού επιπέδου επίθεση. Πώς τα κατάφερνε άραγε αυτή η μικρή φλόγα; Ο προβληματισμός μου ήταν έντονος και ξεκίνησα αμέσως μια διεξοδική έρευνα γύρω από το ζήτημα. Μετά από ατελείωτες ώρες αναζητήσεων και μελέτης, οι προσπάθειές μου κατέληξαν σε ενδιαφέροντα αποτελέσματα. Το WPAD MiTM attack δεν πέθανε και δεν παροπλίστηκε ποτέ. Είναι αυτό που λέμε και στον πλανήτη μου, alive-and-kicking! Αρκεί να σας πω ότι υπάρχει πλέον και σχετικό module για το Metasploit (`www.Metasploit.com/modules/auxiliary/server/wpad`). Όπως καταλαβαίνετε, η διαπίστωση αυτή οδήγησε σε χαρές και πανηγύρια: Είχε παρουσιαστεί μια ευκαιρία για νέες δοκιμές και πειραματισμούς :)

Η θεωρία

Το Web Proxy Auto Discovery (WPAD) αποτελεί μια τεχνολογία που αξιοποιούν οι web browsers εδώ κι αρκετό καιρό, προκειμένου να ανακαλύπτουν αυτόματα την παρουσία proxy servers και να μαθαίνουν τις σχετικές ρυθμίσεις. Χωρίς να μπούμε σε πολλές λεπτομέρειες, θα πούμε μόνο ότι υπάρχουν τρεις μέθοδοι για τον εντοπισμό των proxy: Αναζήτηση μέσω DHCP, μέσω DNS και, τέλος, μέσω NetBIOS (`http://bit.ly/wpadie`). Έτσι, κάθε φορά που ζητάμε μια διεύθυνση από τον browser αυτός στρέφεται αρχικά στον τοπικό DHCP και του κάνει την εξής ερώτηση: “Ξέρεις που θα βρω κανέναν καλό proxy server, για την τοποθεσία που μου ζητήσανε;” Ουσιαστικά, ο browser ζητάει από τον DHCP να του υποδείξει κά-



Εικόνα 1
Ο ένας από τους δύο πρωταγωνιστές του πειράματός μας: Το ανυπoψίαστο θύμα με τα Windows 7. Στην εικόνα φαίνονται και τα δικτυακά του στοιχεία...

ποιον WPAD server. Αν δεν πάρει απάντηση, ο browser στρέφεται στους DNS και τους κάνει μια παρόμοια ερώτηση: "Γνωρίζετε κανέναν server με το όνομα wpad.<dns-suffix>"; Προφανώς, στη θέση του <dns-suffix> μπαίνει το domain name της τοποθεσίας που ζητήσαμε από το browser. Αν αποτύχει κι αυτή η αναζήτηση, ακολουθεί νέα απόπειρα εντοπισμού του WPAD server, αυτή τη φορά μέσω NetBIOS. Αν και πάλι δεν βρεθεί τίποτα, ο browser συνεχίζει χωρίς proxy (Σ.τ.Ε. Και φαντάζομαι ότι λέει κάτι σαν «Ω γουέλ, γουάτ αρ γιου γκόνα ντου αμπάουτ ιτ;»). Ωστόσο, αν εντοπιστεί κάποιος WPAD server, που συνήθως πρόκειται για τον ίδιο τον proxy, ο browser ζητάει ένα αρχείο (wpad.dat) με τα στοιχεία και τις ρυθμίσεις του proxy (IP, port κ.λπ.). Όλα αυτά είναι απλά και κατανοητά. Κάτι δεν πάει καλά, όμως: NetBIOS εν έτει 2012; Είναι δυνατόν, ένας σύγχρονος browser, αφού ρωτήσει τον DHCP server κι ακολούθως τους name servers, να ασχοληθεί και με το NetBIOS; Κι όμως, είναι.

Όπως αντιλαμβάνεστε, η γνώση των παραπάνω οδηγεί σχεδόν αυτόματα σε μια πονηρή σκέψη: Πόσο δύσκολο θα ήταν να στήσουμε έναν transparent proxy server, με NetBIOS name το WPAD; Το ίδιο μηχάνημα θα μπορούσε να τρέχει κι έναν web server, ώστε να μοιράζει το απαιτούμενο wpad.dat, σε όποιον browser το ζητά. Προφανώς, κάτι τέτοιο δεν είναι καθόλου δύσκολο! Όμως πώς θα εξασφαλίσουμε ότι το web proxy auto discovery είναι ενεργοποιημένο στους υπολογιστές του δικτύου μας; Κανένα πρόβλημα! Είναι ενεργοποιημένο εξ ορισμού :D Με λίγα λόγια, φαίνεται ότι είναι σχετικά εύκολο να εισαγάγουμε τον δικό μας proxy server σε ένα ξένο δίκτυο (π.χ. στο hotspot της αγαπημένης σας καφετέριας ή στο WiFi του αγαπητού σας γείτονα). Με αυτόν τον τρόπο θα μπορούμε να παρακολουθούμε όλη τη δικτυακή κίνηση!

Στην πράξη

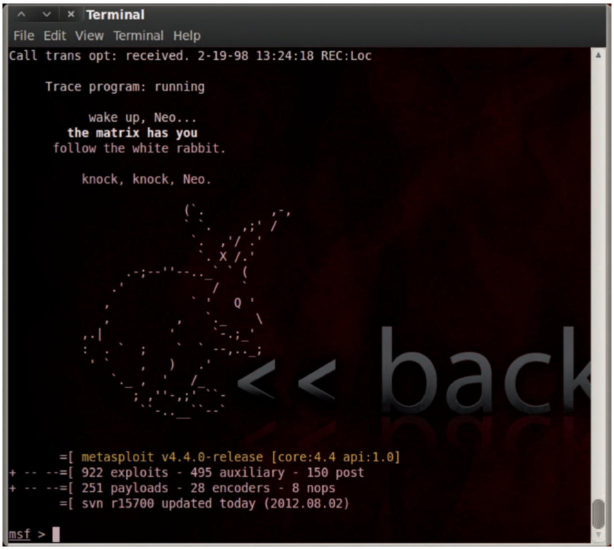
Στον πλανήτη μου, όταν ανακαλύπτουμε κάτι τόσο ενδιαφέρον θέλουμε να το δοκιμάζουμε, να το εφαρμόζουμε στην πράξη και να μαθαίνουμε ακόμα περισσότερα! Το ίδιο θα κάνω κι εγώ λοιπόν και το ίδιο προτείνω να κάνετε κι εσείς. Για τη σχετική δοκιμή θα χρειαστούν δύο virtual machines. Στο ένα εγκατέστησα Windows 7 και είχε τη διεύθυνση 192.168.137.177. Στο άλλο φόρτωσα το BackTrack 5 R2 και είχε τη διεύθυνση 192.168.137.252. Προφανώς, το πρώτο VM θα έπαιζε το ρόλο του θύματος, ενώ το δεύτερο θα είχε το ρόλο του επιτιθέμενου. Αυτό που χρειαζόμαστε από το BackTrack είναι δύο modules του Metasploit (ένα για το NetBIOS spoofing κι ένα για το σερβίρισμα του wpad.dat), καθώς και το Burp Suite (intercepting proxy). Πριν στήσουμε την παγίδα, δημιουργώντας τον ύπουλο proxy server, πρέπει να κάνουμε μια πλήρη ενημέρωση του BackTrack:

```
root@bt:~# apt-get update
```

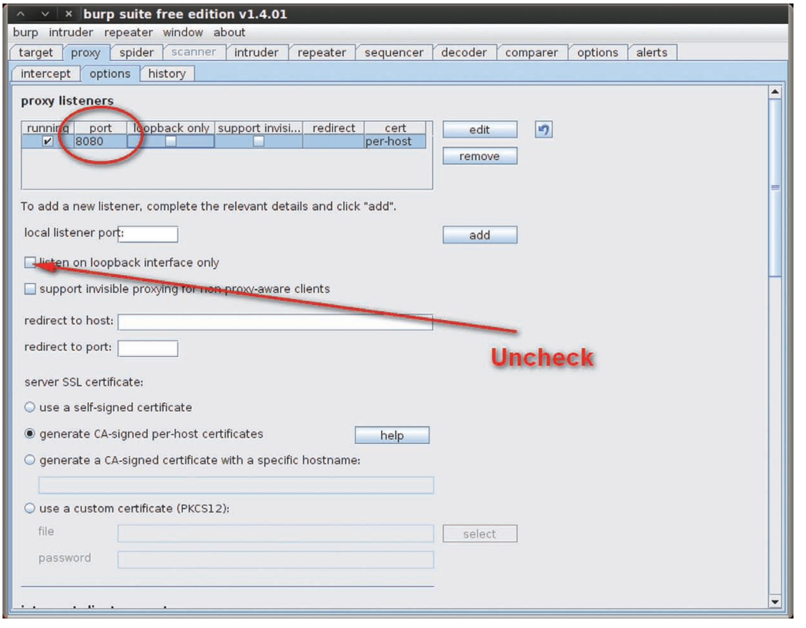
Κι αμέσως μετά…

```
root@bt:~# apt-get upgrade
```

Εικόνα 2
Ο επιτιθέμενος έτρεξε το Metasploit και το framework τον καλωσόρισε με ένα από τα ομορφότερα banner του (follow the whiter rabbit)! Επίσης, φαίνεται ότι το Metasploit έχει ενημερωθεί και αναβαθμιστεί στην εκδοχή 15700. Τη στιγμή που γράφονται αυτές οι γραμμές, πρόκειται για την πλέον πρόσφατη έκδοση του Metasploit…



Εικόνα 3
Οι απαραίτητες ρυθμίσεις για τον proxy server του burp suite…



Στην περίπτωση που έχετε εγκαταστήσει το Metasploit σε κάποια άλλη διανομή, φροντίστε να αναβαθμίσετε τουλάχιστον το ίδιο το framework. Για να πετύχετε κάτι τέτοιο εκτελέστε το ακόλουθο:

```
root@bt:~# msfupdate
```

Έτσι, θα ξεκινήσει η διαδικασία αναβάθμισης του Metasploit, στο τέλος της οποίας θα έχετε φτάσει στο πιο πρόσφατο revision…

```
[*]
[*] Attempting to update the Metasploit Framework...
[*]
...
Updated to revision 15700.
```

Παρεμπιπτόντως, το παραπάνω βήμα μπορούν να κάνουν κι όσοι έχουν BackTrack αλλά βαριούνται να αναβαθμίσουν ολόκληρο το σύστημα. Πριν προχωρήσουμε στο πείραμα καθαυτό, πρέπει να φροντίσουμε κάτι ακόμα. Το σερβίρισμα του wpad.dat θα πραγματοποιείται από ένα ειδικό module του Metasploit. Επομένως, πρέπει να κλείσουμε τον Apache web server (τρέχει εξ ορισμού στο BackTrack), ώστε να μην υπάρξει κάποιο conflict:

```
root@bt:~# service apache2 stop
 * Stopping web server apache2  ... waiting . [ OK ]
```

Όσοι χρησιμοποιούν κάποια άλλη διανομή, πρέπει επίσης να κλείσουν τον Apache -- ή οποιονδήποτε άλλο web server ενδέχεται να είναι ενεργός.

Στήσιμο της παγίδας

Μετά την απαραίτητη προετοιμασία, μπορούμε να ξεκινήσουμε το Metasploit:

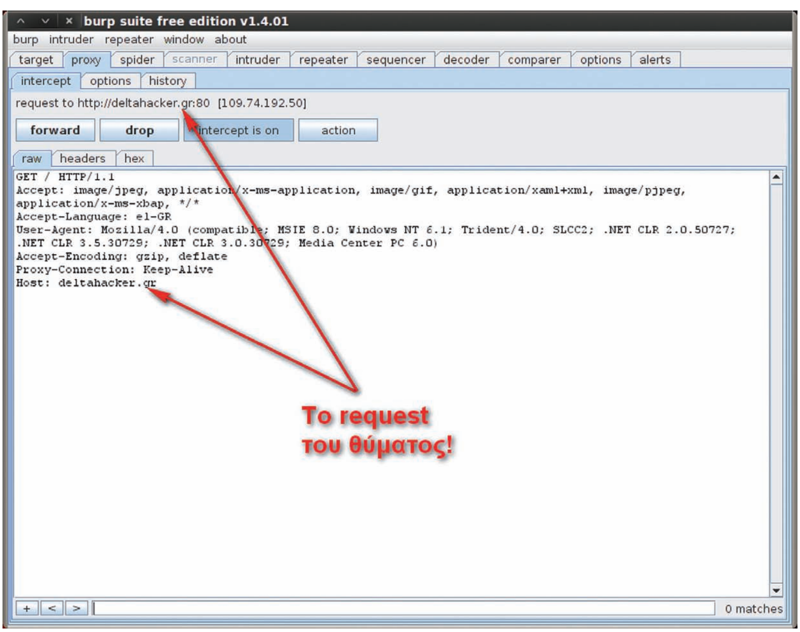


Εικόνα 4
Η περιήγηση στο web ξεκινά εντελώς τυχαία, με ένα login στο forum του deltahacker.gr...

```
root@bt:~# msfconsole
```

Και κάπως έτσι θα μας χαιρετίσει ένα από τα πανέμορφα banners του Metasploit (στη δική μας περίπτωση εμφανίστηκε το "follow the white rabbit", από το Matrix). Θα ενημερωθούμε για την πρόσφατη αναβάθμιση του προγράμματος και θα εμφανιστεί η γραμμή εντολών του:

```
          =[ Metasploit v4.4.0-release [core:4.4 api:1.0]
+ -- --=[ 922 exploits - 495 auxiliary - 150 post
+ -- --=[ 251 payloads - 28 encoders - 8 nops
       =[ svn r15700 updated today (2012.08.02)
msf >
```



Εικόνα 5
WPAD MiTM attack is alive and kicking! Ζητώ συγνώμη, αλλά όταν ξεκίνησα αυτή τη λεζάντα ο μεταφραστής άρχισε να βγάζει καπνούς και σταμάτησε να λειτουργεί :S

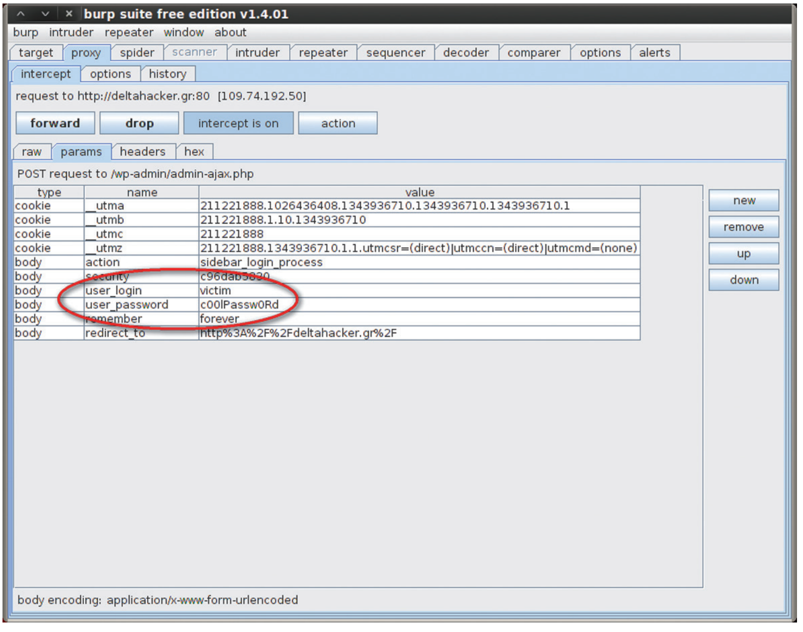
Αρχικά, πρέπει να φορτώσουμε το module για το NetBIOS spoofing. Αμέσως μετά μπορούμε να δούμε τις ρυθμίσεις του, για να ελέγξουμε αν και ποια θα πρέπει να τροποποιήσουμε.

```
msf > use auxiliary/spoof/nbns/nbns_response
msf  auxiliary(nbns_response) > show options
Module options (auxiliary/spoof/nbns/nbns_response):
  Name       Current Setting  Required  Description
  ----       ---------------  --------  -----------
  INTERFACE                   no        The name of the interface
  REGEX      .*               yes       Regex applied to the NB Name to determine
                                           if spoofed reply is sent
  SPOOFIP    127.0.0.1        yes       IP address with which to poison responses
  TIMEOUT    500              yes       The number of seconds to wait for new
                                           data
```

Είναι φανερό, νομίζω, ότι πρέπει να θέσουμε στο REGEX την τιμή WPAD και στο SPOOFIP τη διεύθυνση του server μας. Για σκοπό αυτό, εκτελούμε τα ακόλουθα:

```
msf  auxiliary(nbns_response) > set regex WPAD
regex => WPAD
msf  auxiliary(nbns_response) > set spoofip 192.168.137.252
spoofip => 192.168.137.252
msf  auxiliary(nbns_response) > run
[*] Auxiliary module execution completed
[*] NBNS Spoofer started. Listening for NBNS requests...
```

Αυτό ήταν όλο! Από αυτή τη στιγμή και μετά, η εικονική μηχανή με το



Εικόνα 6
Ο proxy server του burp suite έκανε το θαύμα του! Με τη βοήθειά του μάθαμε και το username και το passoword του ανυποψίαστου θύματος!

BackTrack θα διαφημίζεται στο δίκτυο ως WPAD server. Σειρά έχει το ζήτημα με το διαμοιρασμό του wpad.dat. Κατά τα γνωστά, αφού ξεκινήσουμε το σχετικό module, θα ρίξουμε μια ματιά και στις ρυθμίσεις του...

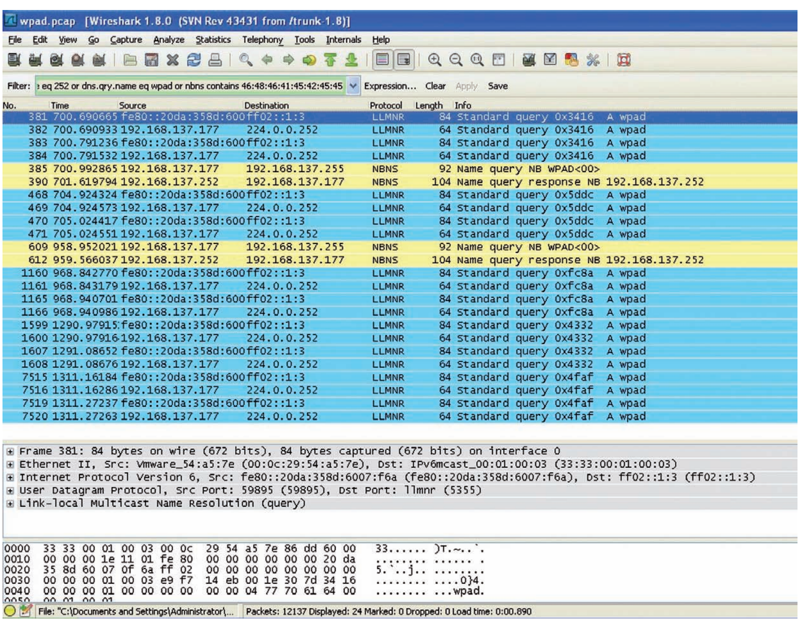
```
msf  auxiliary(nbns_response) > use auxiliary/server/wpad

msf  auxiliary(wpad) > show options

Module options (auxiliary/server/wpad):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   EXCLUDENETMASK  255.255.255.0    yes       Netmask to exclude
   EXCLUDENETWORK  127.0.0.1        yes       Network to exclude
   PROXY           0.0.0.0          yes       Proxy to redirect traffic to
   PROXYPORT       8080             yes       Proxy port
   SRVHOST         0.0.0.0          yes       The local host to listen on. This
                                                must be an address on the local
                                                machine or 0.0.0.0
   SRVPORT         80               yes       The local port to listen on.
   SSL             false            no        Negotiate SSL for incoming
                                                connections
   SSLCert                          no        Path to a custom SSL certificate
                                                (default is randomly generated)
   SSLVersion      SSL3             no        Specify the version of SSL that
                                                should be used (accepted: SSL2,
                                                SSL3, TLS1)
   TYPE            DAT              yes       WPAD/PAC Data File (accepted: DAT,
                                                PAC)
```

Εδώ πρέπει να παρατηρήσουμε την τιμή του PROXYPORT. Πρόκειται για το port στο οποίο θα ακούει ο proxy μας, σύμφωνα με τα λεγόμενα του wpad.dat. Αυτό το μέγεθος δεν χρειάζεται να το αλλάξουμε, αλλά πρέπει



Εικόνα 7
Με τη βοήθεια του κατάλληλου φίλτρου στο Wireshark, κατάφερα να εντοπίσω αμέσως τα ερωτήματα του θύματος και τις απαντήσεις του θύτη, μέσω του πρωτοκόλλου NetBIOS. Το spoofing ήταν απόλυτα επιτυχημένο.

να το σημειώσουμε ή απλά να το θυμόμαστε! Αλλαγή χρειάζεται στην επιλογή PROXY, η οποία καθορίζει τη διεύθυνση IP του ύπουλου proxy server. Προφανώς, πρέπει πάλι να δώσουμε τη διεύθυνση IP του μηχανήματος με το BackTrack...

```
msf  auxiliary(wpad) > set proxy 192.168.137.252
proxy => 192.168.137.252
msf  auxiliary(wpad) > run
[*] Auxiliary module running as background job
[*] Serving wpad.dat on port 80
[*] Using URL: http://0.0.0.0:80/wpad.dat
[*] Local IP: http://192.168.137.252:80/wpad.dat
[*] Server started.
```

Ωραία! Ξεμπερδέψαμε και με το wpad.dat. Τώρα μένει να ενεργοποιήσουμε τον intercepting proxy και ακριβώς γι' αυτό ξεκινάμε το burp suite. Κάντε κλικ στο Applications, πάνω αριστερά, κι ακολουθήστε τη διαδρομή *BackTrack → Vulnerability Assessment → Web Application Assessment → Web Application Proxies → burpsuite*. Μόλις εμφανιστεί το παράθυρο του προγράμματος πηγαίνουμε στην καρτέλα proxy και στην υπό-καρτέλα options. Από εκεί θα τροποποιήσουμε τις ρυθμίσεις του listener. Συγκεκριμένα, πρέπει να ορίσουμε ως listening port εκείνο που θα διαφημίζεται μέσω του wpad.dat. Με άλλα λόγια, πρέπει να ορίσουμε το port που συναντήσαμε λίγο νωρίτερα, σαν PROXYPORT. Μαζί με αυτή την επέμβαση, αποεπιλέγουμε και το "listen on loopback interface only", ώστε ο proxy μας να ακούει σε όλα τα interfaces.

Τι καταφέραμε;

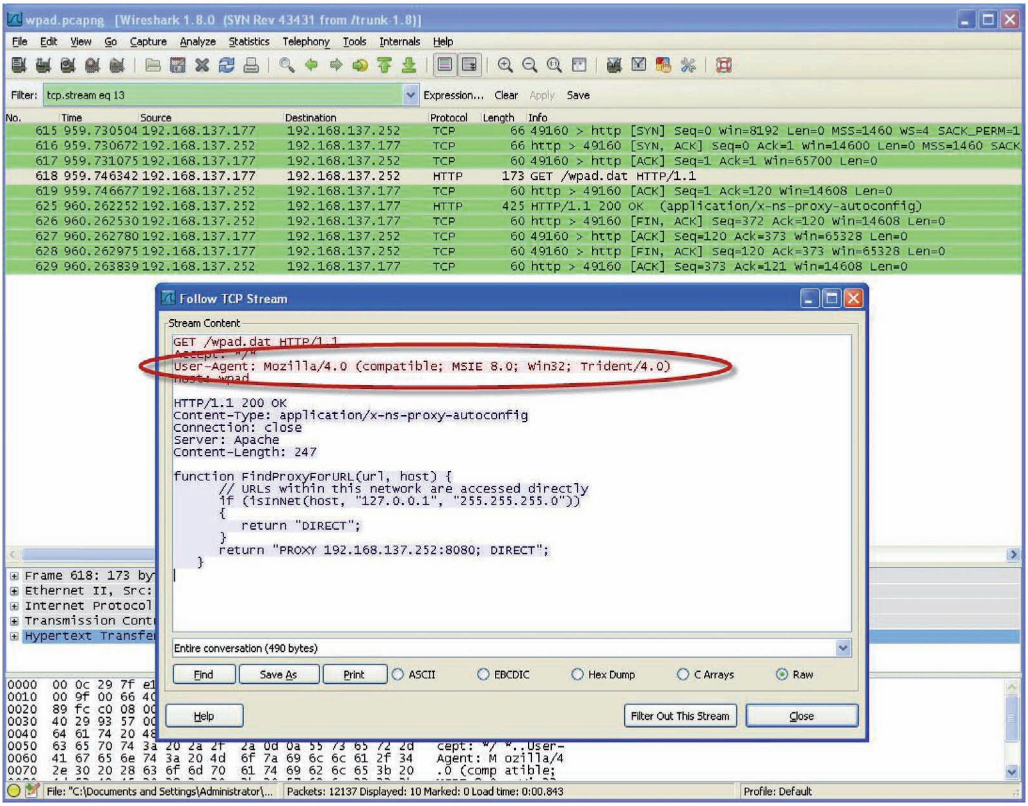
Αυτό ήταν! Πλέον είμαστε σε θέση να παρακολουθούμε όλη την κίνηση του τοπικού δικτύου! Για του λόγου το αληθές, μπορούμε να κάνουμε μερικές δοκιμές από την άλλη εικονική μηχανή, που έχει το ρόλο του θύματος. Η δοκιμή που έκανα εγώ ήταν απλή: Από τον Internet Explorer επισκέφτηκα το http://deltahacker.gr και προσπάθησα να συνδεθώ στο forum, με username το victim και password το c00lPassw0Rd. Με το που ξεκίνησα, στο τερματικό του Metasploit εμφανίστηκε το αίτημα λήψης του wpad.dat...

```
[*] 192.168.137.177  wpad - Request 'GET
[*] 192.168.137.177  wpad - Sending WPAD config ...
[*] 192.168.137.177  wpad - Request 'GET Mozilla/4.0 (compatible; MSIE 8.0; Win32; Trident/4.0)
[*] 192.168.137.177  wpad - Sending WPAD config...
```

Μπορείτε να δείτε τι επακολούθησε, στις εικόνες με το burp suite! Το διαμάντι μας κατάφερε να κλέψει τα πάντα: Τον προορισμό του θύματος, το username που χρησιμοποίησε και φυσικά το password!

Διπλός έλεγχος

Έχοντας κατά νου τη γνωστή διαγαλαξιακή παροιμία «όποιος καεί στο χυλό φυσάει και το γιαούρτι», ήμουν αποφασισμένος να μην αφήσω τίποτα στην τύχη. Αν κάτι πήγαινε στραβά στη δοκιμή, θα ήθελα να ξέρω το γιατί. Σκοπός μου ήταν να παρακολουθήσω τη διαδικασία σε χαμηλό επίπεδο, ώστε να ξέρω ανά πάσα στιγμή τι συνέβαινε. Τελικά η δοκιμή



Εικόνα 8
Παρατηρήστε το TCP Stream με τη μεταφορά του αρχείου wpad.dat. Στις πρώτες γραμμές του wpad.dat αναφέρεται το όνομα του browser του θύματος. Είπατε τίποτα; Το θύμα δεν χρησιμοποιούσε τον Internet Explorer, αλλά τον Firefox!

μου πέτυχε με την πρώτη απόπειρα, αλλά ο μηχανισμός παρακολούθησης που έστησα δεν πήγε χαμένος. Μου προσέφερε το πολυπόθητο insight, που πολύ αγαπάμε στον πλανήτη μου ;)

Κατά το στήσιμο του τοπικού δικτύου, με το VMware Workstation 8, χρειάστηκε να ξεσκονίσω τις γνώσεις μου γύρω από το networking μεταξύ των hosts. Κάπου εκεί όμως έπεσε και το μάτι μου σε μια αναφορά στα λεγόμενα "virtual network hubs", τα οποία δημιουργεί το VMware. "Μια στιγμή βρε παιδιά", αναρωτήθηκα, "είδα τη λέξη hubs ή τα μάτια μου έκαναν αστεράκια;" Στο μυαλό πολλών από εμάς η λέξη "hub» συνεπάγεται ελεύθερο packet sniffing! Έτσι, σκέφτηκα να χρησιμοποιήσω ένα τρίτο Virtual Machine, το οποίο θα παρακολουθούσε την κίνηση σε όλο το Virtual Network Hub. Σκοπός μου ήταν να κάνω packet capture σε όλη συνομιλία μεταξύ του θύματος και του θύτη, για να μπορώ αργότερα να αναλύσω τη διαδικασία. Για το σκοπό αυτό σήκωσα ένα virtual machine με Windows XP, εγκατέστησα το Wireshark και το ξεκίνησα.

Όπως ανέφερα και πριν, η δοκιμή μου στέφθηκε με απόλυτη επιτυχία. Ωστόσο, το Wireshark είχε δημιουργήσει ένα αρχείο καταγραφής με όλο το network traffic. Δεν θα μπορούσα να το αγνοήσω, ούτε να το απορρίψω σαν κάτι το εντελώς άχρηστο. Έτσι, άρχισα να το ψαχουλεύω, για να επιβεβαιώσω τουλάχιστον όσα ήξερα από τη θεωρία. Αρχικά, σκέφτηκα να τσεκάρω την αποτελεσματικότητα του NetBIOS spoofing. Για το σκοπό αυτό, χρησιμοποίησα στο Wireshark το ακόλουθο φίλτρο:

```
bootp.option.type eq 252 or dns.qry.name eq wpad or nbns
contains 46:48:46:41:45:42:45:45
```

Το συγκεκριμένο φίλτρο πιάνει όλες τις μεθόδους proxy detection, τσεκάροντας τα σχετικά αιτήματα προς DHCP, DNS και NetBIOS. Το τι ανακάλυψα εφαρμόζοντας το συγκεκριμένο φίλτρο φαίνεται στην εικόνα 7. Όπως βλέπετε, ο υπολογιστής θύμα (192.168.137.177) ζήτησε από το δίκτυο πληροφορίες μέσω NetBIOS, για τον υπολογιστή με όνομα WPAD. Τότε, έσπευσε να του απαντήσει το μηχάνημα με το BackTrack (192.168.137.252). Το NetBIOS spoofing είχε λειτουργήσει άψογα! Αμέσως μετά, σκέφτηκα να τσεκάρω τον τρόπο διακίνησης του wpad.dat. Από τη θεωρία, γνώριζα ότι το εν λόγω αρχείο διακινείται μέσω αιτημάτων http. Έτσι, χρησιμοποιώντας σαν φίλτρο το http, εντόπισα ένα αίτημα HTTP GET για το wpad.dat. Στη συνέχεια, πραγματοποιώντας το λεγόμενο follow TCP stream, εμφανίστηκε μπροστά μου ολόκληρη η συνομιλία μεταξύ του browser του θύματος και του web server του BackTrack. Παρεμπιπτόντως, μαζί με το request και το αντίστοιχο reply, είδα και το περιεχόμενο του wpad.dat. Το κατάλληλα κατασκευασμένο wpad.dat υποδείκνυε στον browser του θύματος το δικό μου proxy και τον ενημέρωνε για το σχετικό port (8080). Μετά από όλα αυτά, δεν είχα μόνο τη χαρά της επιτυχίας αλλά κι εκείνη της βαθιάς γνώσης. Είχα γνωρίσει το μηχανισμό της επίθεσης με κάθε λεπτομέρεια!

Κίνδυνοι και αντίμετρα

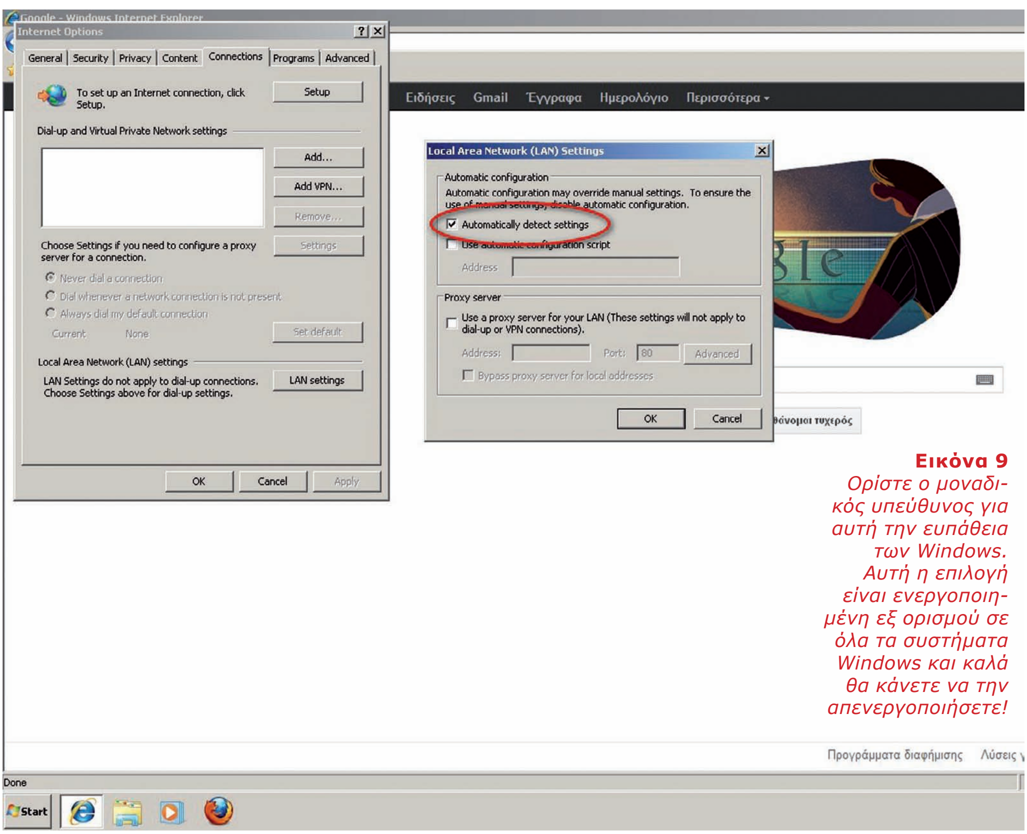
Και κάπως έτσι, το worm με το όνομα flame καταφέρνει να διαδίδεται μέσω μιας ευπάθειας που ισχύει από το 2007! Ήρθε λοιπόν η ώρα να γνωρίσουμε τον αποκλειστικό υπεύθυνο. Ναι, καλά διαβάσατε. Ο υπεύθυνος είναι ένας και μόνο. Πρόκειται για την επιλογή Automatically detect settings του Internet Explorer, η οποία μάλιστα είναι ενεργοποιημένη εξ ορισμού! Σε αυτό το σημείο ακούω αρκετούς από εσάς να δηλώνετε ανακουφισμένοι, επειδή δεν χρησιμοποιείτε Windows ή, πολύ περισσότερο, γιατί αποφεύγετε τον Internet Explorer. Όσοι ανήκετε σε αυτή την κατηγορία μπορείτε δικαιωματικά να νιώθετε ασφαλέστεροι, αλλά όχι απόλυτα. Σκεφτείτε ότι θα μπορούσε κανείς να αντικαταστήσει το NetBIOS spoofing με ένα κατάλληλο DNS poisoning και να πετύχει το ίδιο αποτέλεσμα. Εξάλλου, ο Firefox ή και κάποιος άλλος browser θα μπορούσε να έχει ρυθμιστεί ώστε να χρησιμοποιεί το proxy discovering του συστήματος. Σ’ αυτή την περίπτωση, θα ήσασταν εξίσου εκτεθειμένοι! Σε γενικές γραμμές, πρέπει να θυμόμαστε ότι πρόκειται για μια επίθεση man in the middle, η οποία εκμεταλλεύεται το μηχανισμό WPAD και για την οποία δεν θα διαμαρτυρηθεί κανένα antivirus! Επίσης, ας μην ξεχνάμε ότι το μεγαλύτερο μερίδιο χρηστών Windows ελέγχει πολύ σπάνια τις ρυθμίσεις του Internet. Τέλος, εκτός από την απενεργοποίηση της σχετικής επιλογής, υπάρχουν δύο ακόμα αντίμετρα.

- Να δημιουργήσουμε μια καταχώριση στο αρχείο hosts, της μορφής “127.0.0.1 wpad”, ώστε κάθε φορά που ενεργοποιείται το auto discovery να ανατρέχει στον εαυτό του και φυσικά να αποτυγχάνει.
- Ένας διαχειριστής δικτύου μπορεί να εφαρμόσει το ίδιο κολπάκι καθολικά. Μπορεί να δημιουργήσει μια εσφαλμένη καταχώρηση στις ρυθμίσεις του DHCP server (option 252) και μια αντίστοιχη εγγραφή στις ρυθμίσεις του DNS. Έτσι, όταν ενεργοποιείται σε κάποιον browser το WPAD, η αναζήτηση θα έχει ασφαλή κατάληξη κι όχι κάποιας μορφής spoofing.

Πρόλογος αντί επιλόγου

Σωστά διαβάσατε! Εγώ μόλις ξεκίνησα και δεν έχω καμία διάθεση να σταματήσω, ενώ εσείς διαβάζετε τον πρόλογο των μελλοντικών μου γραπτών. Μπορεί το συγκεκριμένο άρθρο να τέλειωσε, αλλά τώρα που έμαθα να χειρίζομαι τη συσκευή μετάφρασης στα γήινα δεν σκοπεύω να σταματήσω το γράψιμο. Ο εξωγήινος της διπλανής πόρτας ήρθε για να μείνει και να μοιραστεί μαζί σας όλες τις αγαπημένες του δραστηριότητες. Εσείς από τη μεριά σας μη διστάσετε ούτε στιγμή να κάνετε ερωτήσεις, σχόλια και παρατηρήσεις. Hit me hard και με κάθε μέσο που σας προσφέρει το περιοδικό (forum στο site, Facebook fan page, Twitter κ.λπ.). Να ξέρετε ότι εμείς --εκεί, στο outer space-- είμαστε καλόβολοι, εντελώς ακομπλεξάριστοι και (αυτό δεν το μεταφράζει καλά η συσκευή μου) διαπλανιτιστές[1]. Αγαπάμε την εποικοδομητική συζήτηση με fellow hackers, ακόμα κι αν αυτοί προέρχονται από κάποιον ξένο, μακρινό κι εξωτικό πλανήτη, όπως η Γη.

Greetings earthlings!



Εικόνα 9
Ορίστε ο μοναδικός υπεύθυνος για αυτή την ευπάθεια των Windows. Αυτή η επιλογή είναι ενεργοποιημένη εξ ορισμού σε όλα τα συστήματα Windows και καλά θα κάνετε να την απενεργοποιήσετε!

1. Μια παραλλαγή του δικού σας «διεθνιστές». Το αντίθετο του «ρατσιστές», τέλος πάντων, αλλά σε πλανητική κλίμακα.



by GiannouG

Skill: Advanced

Tags: dd-wrt, BitTorrent, OpenWrt, custom firmware, package management, file sharing

Μετατρέψτε τον router σας σε torrent downloader!

Το νέο (νέο) μας router είναι πλέον πιο δυνατό από ποτέ. Στο άρθρο του deltaHacker 011 που αρχίζει από τη σελίδα 60 μιλήσαμε για τα after market firmware των routers, αναφέραμε τις μικροδιαφορές μεταξύ τους κι επιλέξαμε το dd-wrt για το δικό μας. Συζητήσαμε για το πώς βρίσκουμε μια έκδοση συμβατή με το hardware που διαθέτουμε, για το πώς την εγκαταθιστούμε εύκολα και τέλος, από ένα ύψος εννέα περίπου χιλιομέτρων, είδαμε όλες τις νέες δυνατότητες και ρυθμίσεις που μας προσφέρονται. Το τι μπορούμε να κάνουμε από εδώ και πέρα είναι το ενδιαφέρον της υπόθεσης και, σε πολύ λίγο, θα καταλάβουμε γιατί η αλλαγή του firmware στον router ήταν ένα από τα καλύτερα πράγματα που κάναμε για το τοπικό μας δίκτυο.

Μέχρι τώρα δεν έχουμε δει καθόλου την πιο εξειδικευμένη πλευρά του νέου firmware, το οποίο, ας μην ξεχνάμε, είναι μια κανονικότατη διανομή Linux. Θα θυμόσαστε ίσως ότι στο προηγούμενο τεύχος δεν ασχοληθήκαμε καθόλου με την καρτέλα Services του web panel. Στην καρτέλα αυτή θα βρούμε από FTP server μέχρι ολόκληρους VPN servers, έτοιμους να μπουν σε λειτουργία με ελάχιστα κλικ. Επίσης, από τη στιγμή που το ρουτεράκι μας έχει ουσιαστικά μετατραπεί σε ένα κανονικότατο Linux box, σίγουρα μπορούμε να τρέχουμε κι "εξωτερικά" προγράμματα σ' αυτό. Αλλά δεν έχει πολύ νόημα να κάνουμε καταγραφή λειτουργιών και δυνατοτήτων. Αντίθετα, αποφασίσαμε να σας δείξουμε στην πράξη μερικές από τις νέες δυνατότητες που μας προσφέρονται, μετατρέποντας το μικρό μας router σε έναν κανονικότατο torrent downloader. Αυτή η λειτουργία μάλλον δεν ήταν στα σχέδια του κατασκευαστή, μολαταύτα κάτι μας λέει ότι θα τη βρείτε εξαιρετικά χρήσιμη :) Ο downloader θα ελέγχεται μέσω web panel και, φυσικά, τα κατεβασμένα αρχεία θα τα παίρνουμε μέσω FTP ή SMB.

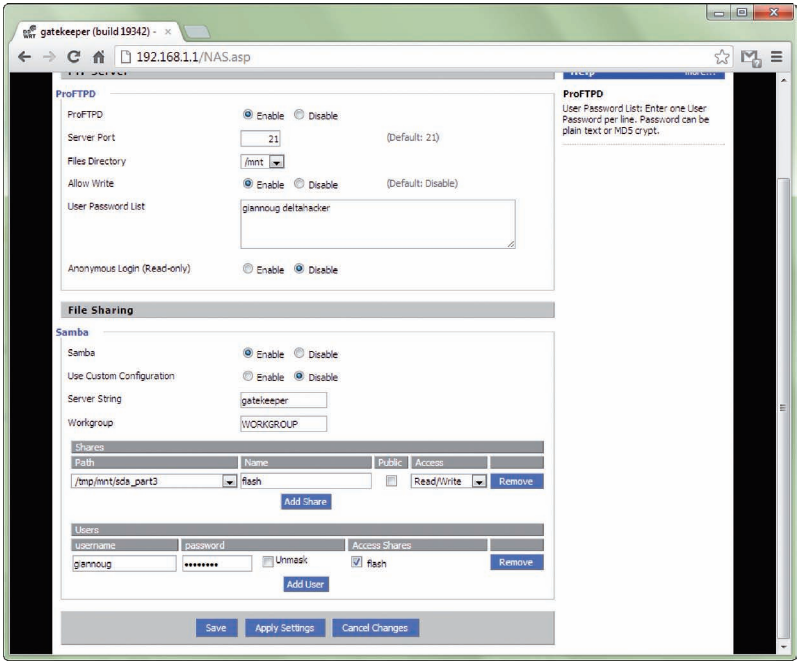
Ρύθμιση και ενεργοποίηση ProFTPd και SAMBA

Το dd-wrt από μόνο του υποστηρίζει αρκετές δικτυακές υπηρεσίες. Μπορούμε π.χ. να ενεργοποιήσουμε τοπικό DNS server ή να ζητήσουμε από τον router να δουλεύει ως OpenVPN server και *ταυτόχρονα* ως client. Ας επικεντρωθούμε όμως στις δυνατότητες που θα επιτρέψουν στον router να λειτουργεί κι ως ένας ικανότατος torrent downloader. Για τη συνέχεια, υποθέτουμε ότι η συσκευή διαθέτει θύρα USB.

Στην καρτέλα USB, λοιπόν, υπάρχουν οι ρυθμίσεις για τη θύρα USB της συσκευής. (Σ.τ.Ε. Duh!) Φυσικά, αν δεν χρειαζόμαστε την αντίστοιχη υποστήριξη, απλά επιλέγουμε το Disable. (Σ.τ.Ε. δις.) Για το σκοπό μας όμως θα φροντίσουμε ώστε να ενεργοποιήσουμε τόσο το Core USB Support,

Ίσως ένα απ' τα πιο ενδιαφέροντα μενού στο dd-wrt: Ενσωματωμένη υποστήριξη VPN!





Το router μας είναι πανέτοιμο και μοιράζεται μέσω FTP και SMB όλα τα αρχεία από το flash disk μας. Εντάξει, για την ώρα δεν κάνει κάτι το συναρπαστικό. Για την ώρα.

όσο και το USB Storage Support. Επιπλέον, ενεργοποιούμε το Automatic Drive Mount και ορίζουμε ως σημείο προσάρτησης (mount point) το /mnt. Χαμηλότερα, στο πεδίο Disk Info, εμφανίζονται πληροφορίες για τις συσκευές αποθήκευσης USB που συνδέουμε πάνω στον router. Θα επιστρέψουμε ξανά σε αυτή την καρτέλα. Για την ώρα, μπορούμε να συνδέσουμε στη συσκευή ένα USB flash disk και να παρατηρήσουμε τι θα γίνει.

Η καρτέλα NAS, όπως προδίδει και το όνομα, περιέχει υπηρεσίες που μετατρέπουν το router μας σε ένα (απλό) network attached storage. Εδώ μπορούμε να ενεργοποιήσουμε και να ρυθμίσουμε τους δύο servers που μας ενδιαφέρουν, συγκεκριμένα τον ProFTPD για το πρωτόκολλο FTP και τη SAMBA για το SMB[1].

Αρχικά ενεργοποιούμε τον ProFTPd, απλά πατώντας το Enable. Θα χρειαστεί να του υποδείξουμε πού βρίσκονται τα αρχεία μας. Είχαμε επιλέξει πριν λίγο ο USB δίσκος να προσαρτάται στο /mnt, οπότε αυτή τη θέση υποδεικνύουμε στον ProFTPd. Επειδή θέλουμε όχι μόνο να παίρνουμε αλλά και να ανεβάζουμε αρχεία μέσω FTP, ενεργοποιούμε και το Allow Write. Τέλος, στο πεδίο User Password List ορίζουμε τους χρήστες οι οποίοι επιθυμούμε να έχουν πρόσβαση. Θα χρειαστεί να δώσουμε ένα χρήστη ανά γραμμή, ξεχωρίζοντας το username από το password με ένα κενό. Παράδειγμα:

```
giannoug lovesjustinbieber
sub0 secretlyloveswindows
```

Αν τώρα δεν μας ενδιαφέρουν οι χρήστες και θέλουμε ο καθένας να μπαίνει στον FTP server μόνο για να παίρνει αρχεία, απλά ενεργοποιούμε το Anonymous Login (Read-only). Αυτό βέβαια προϋποθέτει ότι εμπιστευ-

1. Με το SMB μπορούμε να στέλνουμε και να παίρνουμε αρχεία και φακέλους κατευθείαν μέσα από τα Windows ή το OS X, χωρίς την εγκατάσταση κάποιου έξτρα προγράμματος.

όμαστε όλους τους χρήστες στο τοπικό δίκτυο κι ότι ο server μας δεν φαίνεται από το Internet.

Ήρθε τώρα η σειρά του SAMBA server. Ξεκινάμε επιλέγοντας το Enable, δίνουμε ένα όνομα στο Server String και πληκτρολογούμε το workgroup στο οποίο θέλουμε να συμμετέχει. Παρεμπιπτόντως, τα πιο συνηθισμένα ονόματα είναι WORKGROUP και MSHOME -- μιλάμε για τρελή πρωτοτυπία! Αν πάντως δεν είμαστε σίγουροι για το όνομα του workgroup στο τοπικό μας δίκτυο, σε κάποιο Windows box μπορούμε ν' ανοίξουμε ένα command prompt window και να δώσουμε

```
net config workstation
```

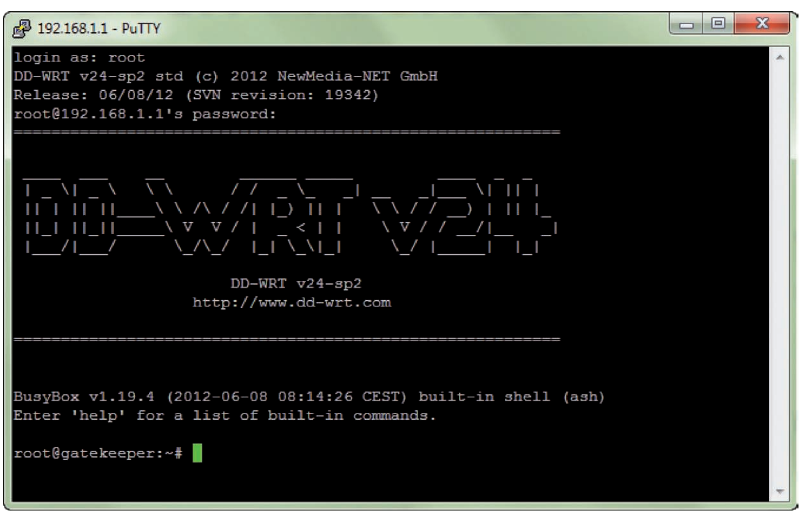
(προσέξτε τη γραμμή Workstation domain.) Ακολουθεί η ρύθμιση του SAMBA server και, πιο συγκεκριμένα, των φακέλων που θα μοιράζει. Στο πρώτο πεδίο δίνουμε το δίσκο μας. Θα πρέπει να υπάρχει μια μόνο επιλογή, αφού έχουμε συνδεδεμένο μόνον έναν δίσκο πάνω τον οποίο και διαλέγουμε. Στο επόμενο πεδίο γράφουμε ένα όνομα, για παράδειγμα Stuff, κι αποεπιλέγουμε το Public. Στον πίνακα ορίζουμε τους χρήστες. Δίνουμε ένα username και password κι επιλέγουμε τον φάκελο που δημιουργήσαμε πριν. Είμαστε πλέον έτοιμοι κι όλα είναι σωστά. Πατάμε στο Apply Settings για να αποθηκευτούν κι ενεργοποιηθούν οι ρυθμίσεις.

Μπορούμε τώρα να δοκιμάσουμε να συνδεθούμε στον router, με τον αγαπημένο μας FTP client και τα στοιχεία που δώσαμε! Επίσης, το ρουτεράκι θα φαίνεται στο δίκτυο του σπιτιού ως ένας υπολογιστής που μοιράζει αρχεία και φακέλους -- χάρη στη SAMBA, φυσικά.

The Linux side of things

Καιρός να δούμε και την Linux πλευρά του router μας. Ξεκινάμε ενεργοποιώντας τον SSH server, ώστε να είμαστε σε θέση να συνδεόμαστε στη συσκευή και μέσω του ομώνυμου πρωτοκόλλου, αποκτώντας πλήρη πρόσβαση στο κέλυφος του λειτουργικού. Πηγαίνουμε στην καρτέλα Services και βρίσκουμε το Secure Shell. Ενεργοποιούμε και πατάμε Apply Settings. Ανοίγουμε τον αγαπημένο μας SSH client[2] και συνδεόμαστε στο

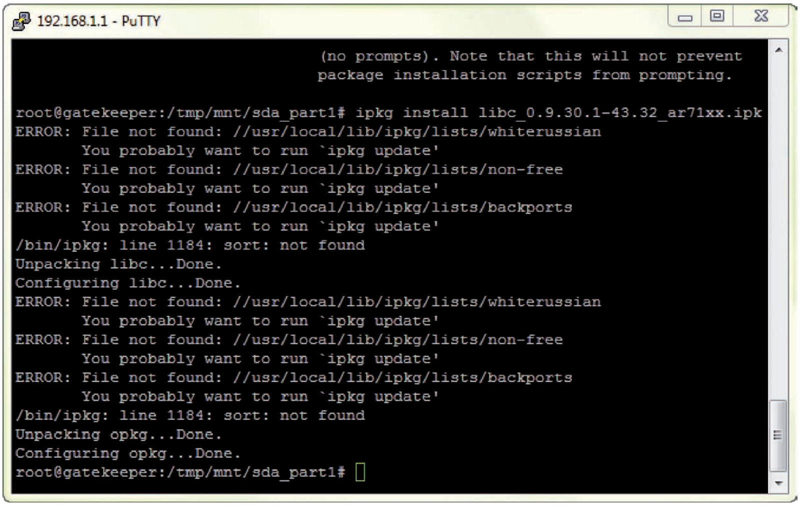
Πολύ Linux θυμίζει εδώ γύρω. Α, ναι, είναι που πράγματι βρισκόμεστε στο Linux. Ποιος θα το περίμενε από το ρουτεράκι μας, ε; :D



2. Το PuTTY για Windows μας καλύπτει απόλυτα!

192.168.1.1 (ή στη διεύθυνση του router, τέλος πάντων). Συνδεόμαστε στον λογαριασμό του root κι ο κωδικός είναι αυτός που δώσαμε την πρώτη φορά που συνδεθήκαμε στη συσκευή.

Το λειτουργικό θα μας καλωσορίσει. Δε θα το κρύψουμε, προς το παρόν δεν έχουμε και πολλά να κάνουμε εδώ. Μπορεί να πρόκειται για ένα πλήρες σύστημα Linux, λείπουν ωστόσο αρκετά σημαντικά κομμάτια που θα το χαρακτήριζαν ως ένα πραγματικά ολοκληρωμένο περιβάλλον. Πώς, π.χ., εγκαθιστούμε πακέτα, όπως πολύ ωραία κι όμορφα κάνουμε σε άλλες διανομές σαν το Ubuntu, το openSuSE και τόσες άλλες;



Η εγκατάσταση του opkg πήγε ρολόι. Μην δίνετε σημασία στα μηνύματα λάθους. Αγνοήστε τα επιδεικτικά και συνεχίστε τη ζωή σας σαν να μην τα είδατε ποτέ. Καταλαβαίνουμε πόσο δύσκολο είναι αυτό για κάποιους, όμως ακούστε μας.

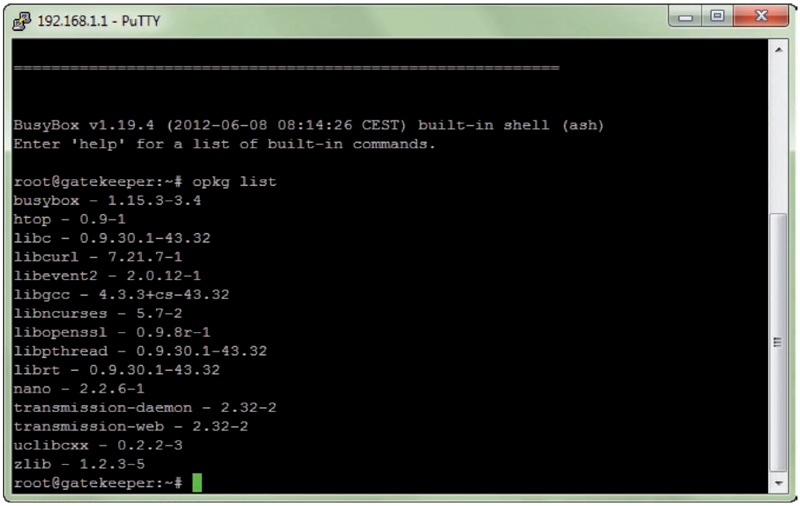
Εγκατάσταση και ρύθμιση Optware

Πριν συνεχίσουμε και στην ουσία απαντήσουμε στο προηγούμενο ερώτημα, σκόπιμο είναι να κάνουμε μια σύντομη, προκαταρκτική παρουσίαση για το λεγόμενο Optware. Ο όρος καθιερώθηκε από το Unslung Linux (`www.nslu2-linux.org/wiki/Main/HomePage`) κι αρχικά αφορούσε σε software για το Linksys NSLU2 (`http://en.wikipedia.org/wiki/NSLU2`). Ακριβώς το ίδιο όνομα, Optware, έχει κι ο σχετικός package manager, ο οποίος δεν ασχολείται με λεπτομέρειες όπως, εχμ, ας πούμε διανομή, αλλά εγκαθιστά εφαρμογές σε ένα συγκεκριμένο μέρος[3] από όπου ο χρήστης μπορεί να τις τρέχει. Το Optware, ενώ ξεκίνησε από το Unslung Linux σύντομα πέρασε και σε άλλες διανομές, φυσικά και σε αυτή του dd-wrt!

Υπάρχει όμως τώρα ένα άλλο πρόβλημα. Τα πακέτα του Optware είναι ήδη μεταγλωττισμένα για επεξεργαστές ARM, ενώ το μικρό μας router είναι από εκείνα με επεξεργαστή MIPS. Τι κάνουμε λοιπόν σε μια τέτοια περίπτωση; Κάνουμε κάτι όχι και τόσο αναμενόμενο, που όμως δουλεύει: Χρησιμοποιούμε τον αντίστοιχο package manager από το αδελφό του dd-wrt πρότζεκτ, ονόματι OpenWrt. Και το κάνουμε γιατί τα πακέτα --τουλάχιστον τα δημοφιλή-- του OpenWrt, είναι μεταγλωττισμένα για περισσότερες από μία αρχιτεκτονικές επεξεργαστών, μεταξύ αυτών και για MIPS. Ας δούμε πώς εργαζόμαστε.

Βήμα 1. Για αρχή χρειαζόμαστε ένα φλασάκι USB ή έναν εξωτερικό σκληρό -- πάντα USB. Για την περίπτωση που εξετάζουμε θέλουμε κυρίως να

3. Συνήθως στο /opt, εξ' ου και το όνομα.



```
192.168.1.1 - PuTTY
============================================================

BusyBox v1.19.4 (2012-06-08 08:14:26 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

root@gatekeeper:~# opkg list
busybox - 1.15.3-3.4
htop - 0.9-1
libc - 0.9.30.1-43.32
libcurl - 7.21.7-1
libevent2 - 2.0.12-1
libgcc - 4.3.3+cs-43.32
libncurses - 5.7-2
libopenssl - 0.9.8r-1
libpthread - 0.9.30.1-43.32
librt - 0.9.30.1-43.32
nano - 2.2.6-1
transmission-daemon - 2.32-2
transmission-web - 2.32-2
uclibcxx - 0.2.2-3
zlib - 1.2.3-5
root@gatekeeper:~#
```

Λίστα με όλα τα εγκατεστημένα πακέτα για την ομαλή λειτουργία του Transmission BitTorrent client, εντός του router μας!

κατεβάζουμε αρχεία, οπότε έχει περισσότερο νόημα η χρήση εξωτερικού σκληρού. Τον δίσκο αυτό πρέπει να τον διαμορφώσουμε κατά Ext3. Αργότερα, θα προσαρτηθεί από το λειτουργικό του router κι εκεί θα εγκαθίστανται τα πακέτα του Optware. Πέρα από αυτά, θα αποθηκεύονται και τα αρχεία που θα κατεβάζουμε/ανεβάζουμε. Η διαμόρφωση του δίσκου κατά Ext3 είναι δυνατόν να γίνει με το mkfs του Linux ή με τη βοήθεια ενός εργαλείου σαν το GParted Live (http://gparted.sourceforge.net/livecd.php).

Βήμα 2. Βάζουμε τον δίσκο πάνω στον router και συνδεόμαστε στη συσκευή μέσω SSH. Φτιάχνουμε τους απαραίτητους φακέλους κι εγκαθιστούμε τα πρώτα μας αρχεία. Εκτελούμε καθεμία από τις παρακάτω εντολές. (Ο δίσκος είναι προσαρτημένος κάτω από το /mnt/sda_part1 κι όχι κάτω από το /mnt, όπως θα περίμενε κανείς.)

```
cd /mnt/sda_part1

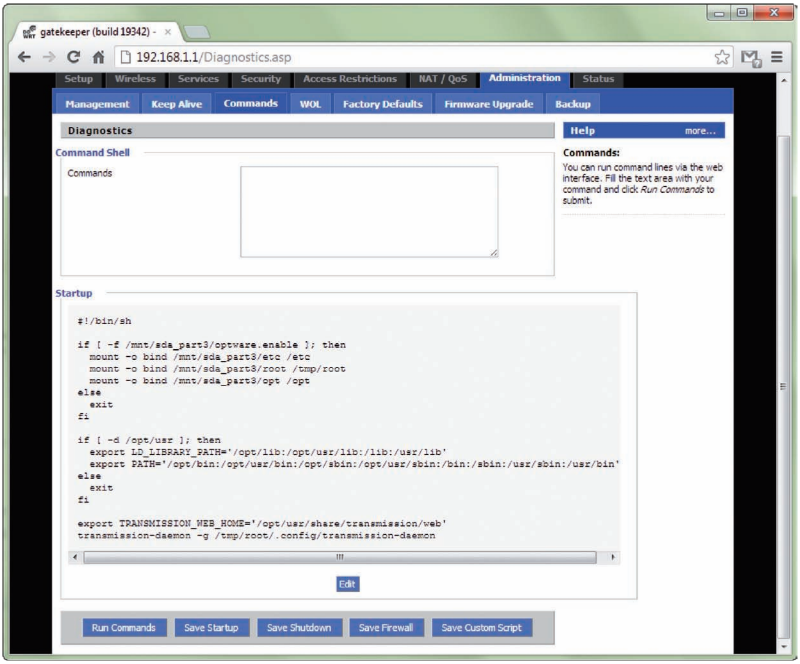
mkdir etc opt root
chmod 755 etc opt root
mkdir opt/lib
chmod 755 opt/lib
touch optware.enable

cp -a /etc/* etc/
mount -o bind etc/ /etc
mount -o bind opt/ /jffs

wget http://downloads.openwrt.org/ \ backfire/10.03.1/ar71xx/
packages/libc_0.9.30.1-43.32_ar71xx.ipk

wget http://downloads.openwrt.org/  \ backfire/10.03.1/ar71xx/
packages/opkg_576-2_ar71xx.ipk

cat > /etc/opkg.conf << EOF
src/gz snapshots http://downloads.openwrt.org/ \
backfire/10.03.1/ar71xx/packages
dest root /opt
dest ram /tmp
lists_dir ext /tmp/var/opkg-lists
EOF
```



Το BASH script που θα εκτελείται κατά την εκκίνηση του router. Αν ο εξωτερικός δίσκος ή το φλασάκι δεν βρεθεί, η εκκίνηση προχωρά ομαλά.

Βήμα 3. Είμαστε σχεδόν έτοιμοι. Πηγαίνουμε στο web panel του router μας. Θα πρέπει να εισαγάγουμε κάποιες εντολές του BASH, οι οποίες θα εκτελούνται κατά την εκκίνηση της συσκευής. Επιλέγουμε Administration → Commands και ρίχνουμε στο κουτί το παρακάτω.

```
#!/bin/sh

if [ -f /mnt/sda_part3/optware.enable ]; then
  mount -o bind /mnt/sda_part1/etc /etc
  mount -o bind /mnt/sda_part1/root /tmp/root
  mount -o bind /mnt/sda_part1/opt /opt
else
  exit
fi

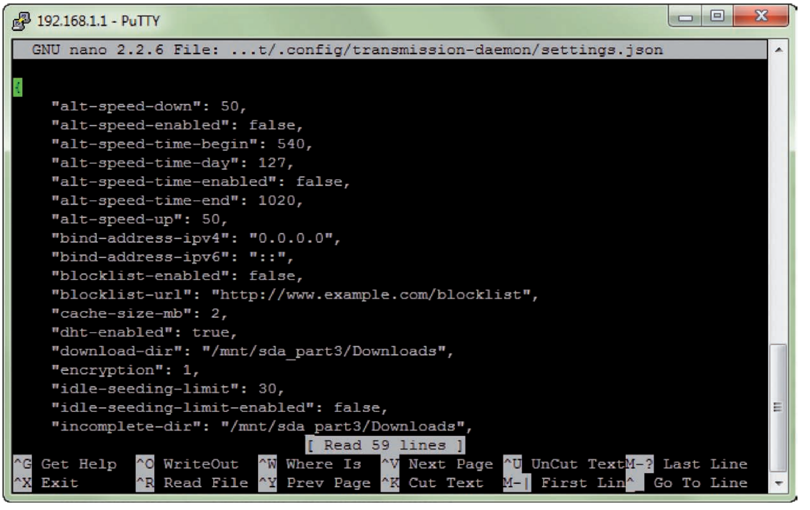
if [ -d /opt/usr ]; then
  export \
LD_LIBRARY_PATH='/opt/lib:/opt/usr/lib:/lib:/usr/lib'
  export \ PATH='/opt/bin:/opt/usr/bin:/opt/sbin:/opt/usr/sbin:/
bin:/sbin:/usr/sbin:/usr/bin'
else
  exit
fi
```

Πατάμε Save Startup κι επανεκκινούμε τη συσκευή, από το *Administration → Setup → Reboot Router*. Μόλις ξεκινήσει συνδεόμαστε και πάλι μέσω SSH. Για να δούμε αν όλα πήγαν καλά, τρέχουμε τον νέο διαχειριστή πακέτων:

```
opkg update
```

Σε περίπτωση λάθους κοιτάμε πάλι όλες τις εντολές και σιγουρευόμαστε πως ο δίσκος μας όντως βρίσκεται στον φάκελο sda_part1. Αν δεν είμα-

Το αρχείο ρυθμίσεων του Transmission δεν είναι ιδιαίτερα πολύπλοκο και περιέχει όλες τις κλασικές ρυθμίσεις ενός BitTorrent client.



στε σίγουροι, μπορούμε να το δούμε από το μενού USB, που είδαμε πριν.

Τέλος, εγκαθιστούμε ορισμένα πακέτα, τα οποία περιέχουν τις βασικές βιβλιοθήκες που χρειάζονται οι εφαρμογές για να τρέχουν:

```
opkg install libc
opkg install libgcc
opkg install uclibcxx
```

Πριν συνεχίσουμε κάντε ένα σύντομο διάλειμμα, για να εγκαταστήσετε τα nano και htop. Γενικά, έχετε υπόψη ότι η σύνταξη του opkg είναι παρόμοια με τη σύνταξη του apt-get, από τις Debian-based διανομές Linux.

Εγκατάσταση Transmission

Όλα ωραία και καλά, έχουμε μετατρέψει έναν μικρό και ταπεινό router σε έναν σχεδόν πλήρη Linux server. Ώρα να κάνουμε και κάτι πιο ενδιαφέρον. Θα εγκαταστήσουμε τον BitTorrent client ονόματι Transmission μαζί με το web panel του! Ξεκινάμε πληκτρολογώντας τα ακόλουθα:

```
opkg update
opkg install transmission-daemon transmission-web
```

Τρέχουμε για λίγο το πρόγραμμα, ίσα για να προλάβει να δημιουργήσει τα απαραίτητα αρχεία ρυθμίσως, κι αμέσως το σταματάμε:

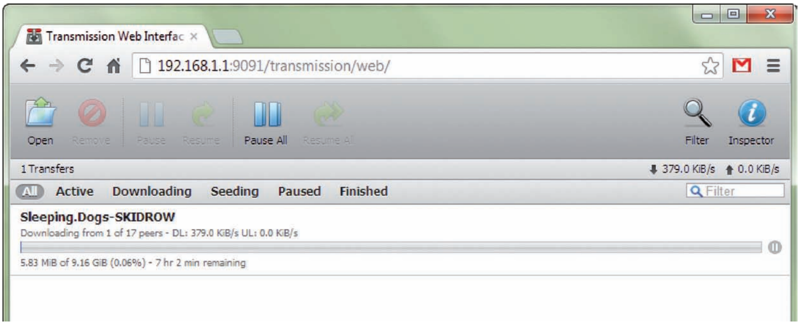
```
transmission-daemon
killall transmission-daemon
```

Θα χρειαστεί να πειράξουμε δύο σημεία σε όλο το σχετικό config. Πρώτα απ' όλα πρέπει να ορίσουμε το πού θα αποθηκεύονται τα αρχεία. Μετά, θα χρειαστεί να επιτρέψουμε και την πρόσβαση στο web panel του Transmission. Ανοίγουμε το αρχείο ρυθμίσεων:

```
nano /tmp/root/.config/transmission-daemon/settings.json
```

Οι αλλαγές που θα χρειαστεί να κάνουμε φαίνονται παρακάτω.

```
"download-dir": "/mnt/sda_part1/Downloads",
[...]
```



*Οι πρώτες δοκιμές στέφθηκαν με επιτυχία! Το router μας πλέον λειτουργεί *και* ως BitTorrent client!*

```
"incomplete-dir": "/mnt/sda_part1/Downloads",
[...]
"rpc-whitelist": "127.0.0.1,192.168.1.*",
```

Αποθηκεύουμε τις αλλαγές στο settings.json κι εγκαταλείπουμε τον editor. Δεν ξεχνάμε να δημιουργήσουμε τον κατάλογο των torrent, όπως τον δηλώσαμε στο αρχείο ρυθμίσεων του Transmission:

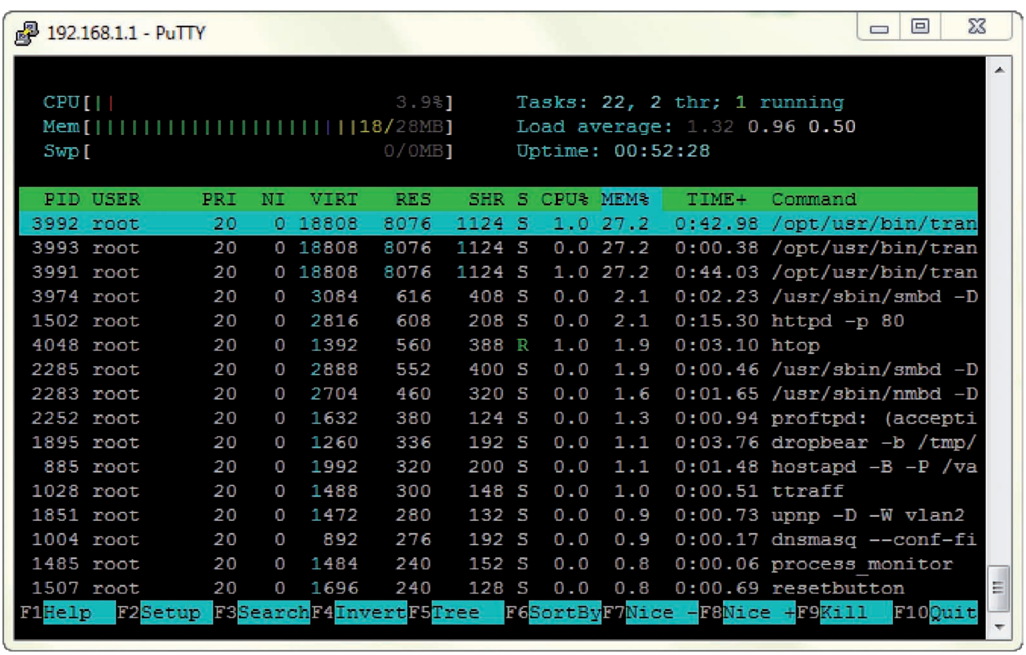
```
mkdir /mnt/sda_part1/Downloads
```

Τέλος, στο web panel του router πηγαίνουμε στο *Administration → Commands* και προσθέτουμε τα παρακάτω, στο τέλος των εντολών που βάλαμε πριν:

```
export TRANSMISSION_WEB_HOME= \ '/opt/usr/share/transmission/web'
transmission-daemon -g /tmp/root/.config/transmission-daemon
```

Ένα τελευταίο restart του router και είμαστε έτοιμοι! Πηγαίνοντας στο 192.168.1.1:9091 με τον αγαπημένο μας web browser, θα δούμε το web panel του Transmission. Τα υπόλοιπα τα αφήνουμε σ' εσάς!

Η αλήθεια είναι ότι η συσκευή ζορίζεται κάπως παραπάνω με το Transmission, τα καταφέρνει μια χαρά όμως!



Python και wxPython:

Ανάπτυξη εφαρμογών σε περιβάλλον γραφικών

Ξέρουμε ότι μόλις διαβάσατε τον τίτλο αισθανθήκατε κάτι μέσα σας να σπάει: Είναι ακόμα νωπές οι μνήμες από το γνωστό μάθημα ΑΕΠΠ (Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον), που μάλλον διδαχθήκατε στο Λύκειο. Ξέρετε, εκείνο που δεν είχε καν... προγραμματιστικό περιβάλλον (εκτός από το χαρτί τις περισσότερες φορές) και που είχε αναγάγει τη δημιουργία προγραμμάτων υπολογισμού κλιμακωτής χρέωσης της ΔΕΗ σε ύψιστο πρόβλημα της πληροφορικής :S

by Sonic



Skill: Intermediate

Tags: Python, wxPython, GUI applications, object oriented programming, objects, events, classes

Θέλουμε να πιστεύουμε πως με την προηγούμενη σειρά άρθρων μας πάνω στην Python και το PyGame, που ξεκίνησε από το deltaHacker 005 (τεύχος Φεβρουαρίου 2012, http://deltahacker.gr/?p=4215), σας αποδείξαμε ότι ο προγραμματισμός δεν χρειάζεται να είναι βαρετός, καθώς κι ότι υπάρχει κάτι περισσότερο από τα προγράμματα κειμένου και κονσόλας. Όχι ότι δεν το γνωρίζατε, δηλαδή, αλλά έτσι όπως διδάσκεται ο προγραμματισμός στο σχολείο είναι πιθανό να σας δημιουργήθηκε η εντύπωση πως το περιβάλλον γραφικών είναι μόνο για τους ueber gurus της πληροφορικής! Για να μην παρεξηγούμαστε, πάντως, θα πούμε εδώ ότι εμάς τουλάχιστον μας αρέσουν (πολύ) τα προγράμματα κονσόλας. Σίγουρα όμως θα πρέπει να κάνουν κάτι πιο ενδιαφέρον από τις βαρετές ρουτίνες του ΑΕΠΠ.

Ίσως να αναρωτιέστε για ποιο λόγο δεν διδάσκεστε προγραμματισμό γράφοντας προγράμματα σε/για περιβάλλον γραφικών ή ακόμη και παιχνίδια, όπως κάναμε στο περιοδικό. Μπορούμε να σκεφτούμε μερικούς λόγους...

- Ο προγραμματισμός σε περιβάλλον κειμένου δεν απαιτεί τη γνώση πολύπλοκων τεχνικών πρόσβασης στην οθόνη και στις συσκευές εισόδου και επιτρέπει να επικεντρώνουμε στη δημιουργία του αλγόριθμου.
- Ο προγραμματισμός σε περιβάλλον γραφικών βασίζεται σε τεχνικές συμβάντων (όπως έχουμε δει στο PyGame) και είναι δύσκολος για κάποιον που δεν γνωρίζει ακόμα τα βασικά της πρώτης του γλώσσας προγραμματισμού.
- Σε πολλές περιπτώσεις, ο προγραμματισμός μια εφαρμογής γραφικών απαιτεί έννοιες και τεχνικές αντικειμενοστρεφούς προγραμματισμού, ενώ τα κλασικά βιβλία στο σχολείο επικεντρώνουν σε διαδικαστικές γλώσσες προγραμματισμού (procedural programming languages, όπως είναι η Pascal).

Και κάπου εδώ ακούμε κάποιους από εσάς να φωνάζουν ότι έχουν γράψει προγράμματα σε Visual Basic (πιθανόν στην παλιά έκδοση, την 6.0). Δεν θέλουμε να αρχίσουμε τις κακίες για τη VB (θα το κάνουμε όμως αν επιμείνετε). Μπορούμε ωστόσο να σας πούμε ένα βασικό χαρακτηριστικό της VB: Κρύβει πάρα πολλές λεπτομέρειες της δημιουργίας μιας εφαρμογής γραφικών πίσω από ένα έτοιμο περιβάλλον ανάπτυξης, το οποίο ουσιαστικά γράφει (αόρατα) κώδικα για εμάς.

Το πρόβλημα με την παραπάνω αντιμετώπιση είναι ότι όποιος έχει γράψει εφαρμογή μόνο στη VB, στην πραγματικότητα δεν ξέρει τι χρειάζεται για να αναπτυχθεί μια εφαρμογή γραφικών. Έχει *μόνο* μια ιδέα από συμβάντα (events), αλλά δεν γνωρίζει πώς ακριβώς καλούνται (λόγω αυτού του κώδικα, που γράφεται αόρατα).

Ο γράφων, θέλοντας στις παλιές καλές εποχές να μάθει τι πραγματικά σημαίνει ανάπτυξη εφαρμογής γραφικών (για Windows, τότε) αγόρασε κάποια βιβλία. Με τρόμο τότε διαπίστωσε ότι το κλασικό, των πέντε γραμμών, πρόγραμμα Hello World στη C, ήταν δύο ολόκληρες σελίδες στην περίφημη Visual C! Κι αυτό γιατί στην Visual C δεν υπάρχει κρυφός κώδικας κι ο προγραμματιστής πρέπει να γνωρίζει τι γίνεται στο παρασκήνιο.

Δεν χρειάζεται όμως να παλεύουμε τον προγραμματισμό εφαρμογών γραφικών, με γλώσσες όπως η C. Γιατί, καθώς φαντάζεστε, είναι πολύ βολικό να αντιμετωπίζουμε όλα αυτά τα καλούδια που μας παρέχει το περιβάλλον γραφικών (παράθυρα, εικονίδια, πεδία κειμένου, λίστες, combo boxes κι όλα τα άλλα widgets, όπως λέγονται) ως αντικείμενα. Η C όμως δεν είναι

αντικειμενοστρεφής γλώσσα και ίσως εκπλαγείτε αν μάθετε ότι το API των Windows έχει γραφτεί σ' αυτήν και *δεν* είναι αντικειμενοστρεφές!

Τι κάνουμε λοιπόν σε μια περίπτωση που θέλουμε να αντιμετωπίσουμε τον προγραμματισμό σε περιβάλλον γραφικών ως αντικειμενοστρεφή, ακόμα και αν εσωτερικά δεν είναι; Πολύ απλά, καταφεύγουμε σε ένα GUI toolkit! Η ίδια η Microsoft παρουσίασε τότε το MFC (Microsoft Foundation Classes), το οποίο ήταν ένα wrapper library: Έκανε τον προγραμματισμό εφαρμογών γραφικών σε Windows αντικειμενοστρεφή, προσθέτοντας ένα ενδιάμεσο layer το οποίο μετέφραζε τα αντικείμενα του MFC στις κανονικές κλήσεις του API των Windows. Έτσι, μπορούσε κάποιος να γράψει την εφαρμογή του στη Visual C++.

Βέβαια εδώ δεν θα σας μιλήσουμε για C, C++ ή Windows (αν και τα προγράμματα μας θα τρέχουν μια χαρά *και* σε αυτό το λειτουργικό), αφού για άλλη μια φορά το όπλο μας θα είναι η Python.

Python και προγραμματισμός γραφικών

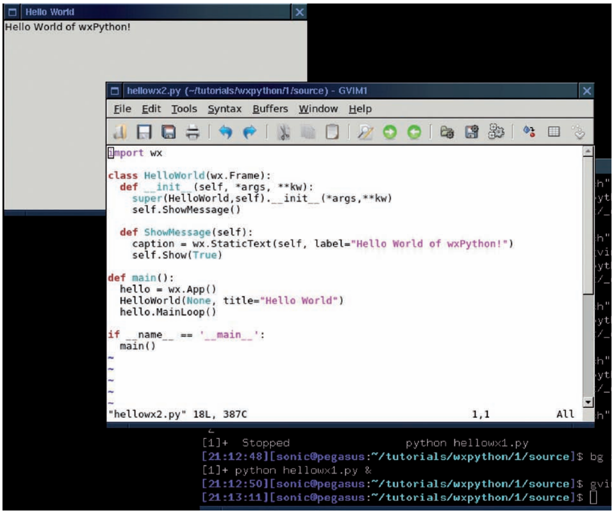
Ό,τι έκανε η MS με το MFC γρήγορα έγινε και σε άλλα λειτουργικά συστήματα και γλώσσες: Ο αντικειμενοστρεφής προγραμματισμός βολεύει ιδιαίτερα, όταν γράφουμε εφαρμογές γραφικών. Εδώ όμως έχουμε ένα άλλο θέμα: η Python είναι μια γλώσσα διαπλατφορμική (cross-platform). Τα προγράμματα που γράφουμε σε αυτήν θα πρέπει να εκτελούνται παντού -- ή τουλάχιστον σε όποια πλατφόρμα η γλώσσα είναι διαθέσιμη και φυσικά εγκατεστημένη. Αυτό δημιουργεί το παράδοξο να πρέπει διαφορετικά περιβάλλοντα γραφικών (τα οποία ενδεχομένως εσωτερικά λειτουργούν διαφορετικά) να προγραμματίζονται με τον ίδιο τρόπο :S

Και κάπου εδώ έρχεται να μας σώσει το wxPython.

Εντάξει, το ξέρουμε ότι δεν είναι εντυπωσιακό, αλλά είναι το πρώτο σας πρόγραμμα σε wxPython. Αν θέλαμε να είμαστε πιο σωστοί βέβαια θα δημιουργούσαμε ένα device context και θα χρησιμοποιούσαμε την DrawText, για να βάλουμε το κείμενο (το StaticText widget είναι ένα γρήγορο κι εύκολο hack). Τώρα που είδατε τα Device Contexts στο πρόγραμμα Drawing, τι λέτε; Μπορείτε να το φτιάξετε; Δείτε κι εδώ: http://wxpython.org/docs/api/wx.DC-class.html#DrawText

Η βασική ιδέα του wxPython

Το wxPython είναι μια διαπλατφορμική βιβλιοθήκη εργαλείων (toolkit) για δημιουργία εφαρμογών γραφικών (GUI applications). Αυτό σημαίνει ότι η βιβλιοθήκη αυτή έχει φτιαχτεί ώστε να προσφέρει μια σταθερή διεπαφή προγραμματισμού (programming interface) προς εμάς, τους χρήστες της, αλλά ταυτόχρονα μεταφράζει κατάλληλα τον κώδικά μας, ανάλογα με το λειτουργικό κάτω από το οποίο εκτελείται. Με απλά λόγια, αυτό σημαίνει τα ακόλουθα:



- Αν γράψετε μια εφαρμογή σε Python και wxPython, θα μπορείτε να την τρέχετε σε Linux Windows, OS X και φυσικά BSD (που θα έγραφα το άρθρο αλλιώς, μου λέτε;)
- Όταν εκτελέσετε την εφαρμογή σε καθένα από αυτά τα λειτουργικά συστήματα, η εμφάνιση της θα είναι φυσιολογική. Στα Windows, π.χ., το παράθυρο, τα εικονίδια κ.ο.κ. δεν θα διαφέρουν σε αίσθηση και λειτουργία από

τις άλλες κλασικές εφαρμογές Windows. Το ίδιο φυσιολογικό θα φαίνεται και στα υπόλοιπα λειτουργικά: Η εφαρμογή σε καθένα θα μοιάζει με αντίστοιχη native.

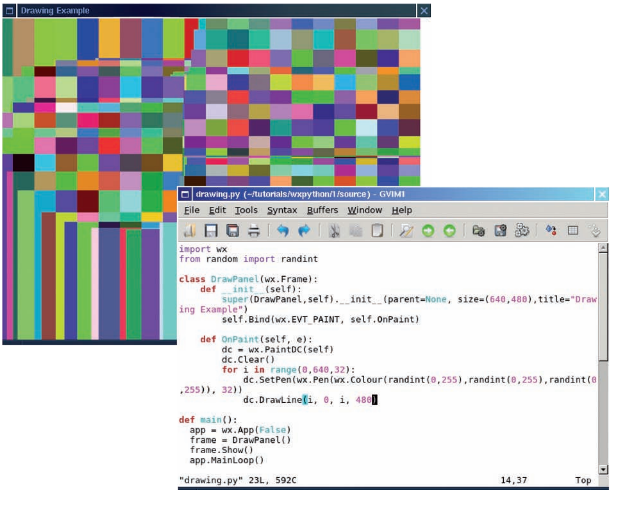
Αν πιστεύετε ότι αυτό το τελευταίο δεν είναι σημαντικό, δοκιμάστε να τρέξετε το IDLE (τον Python editor) σε διάφορα λειτουργικά και θα εκπλαγείτε. Βλέπετε, το IDLE δεν είναι γραμμένο σε wxPython αλλά σε μια άλλη βιβλιοθήκη, το TKinter, η οποία δεν ακολουθεί τη φιλοσοφία της πρώτης. Έτσι, στα Windows το IDLE φαίνεται τελείως παράταιρο, καθώς το παράθυρο και το μενού δείχνει σαν να έχει έρθει από κάποια διανομή Linux -- και μάλιστα της παλιάς εποχής!

Το wxPython, αλλά από μέσα

Δεν χρειάζεται προφανώς να ξέρετε τα πάντα γύρω από την εσωτερική λειτουργία του wxPython, για να γράψετε τα πρώτα σας προγράμματα σε αυτό. Είναι όμως χρήσιμο να ξέρετε ότι το wxPython είναι ένας wrapper για την βιβλιοθήκη wxWidgets. Το πρόβλημα είναι ότι η βιβλιοθήκη wxWidgets είναι γραμμένη σε C++ και εμείς θέλουμε Python. Η wxPython αποτελεί μια ενδιάμεση διεπαφή: Με απλά λόγια, μεταφράζει κάθε εντολή στην αντίστοιχη που καταλαβαίνει η wxWidgets.

- Η **wxPython** αποτελείται από πέντε βασικά αρθρώματα ή modules, αν προτιμάτε:
- Τα **Controls**, τα οποία περιέχουν τα πιο κοινά widgets που χρησιμοποιούν οι εφαρμογές γραφικών.
- Το **Core**, το οποίο περιέχει βασικές κλάσεις που χρησιμοποιούνται σε όλα τα άλλα αρθρώματα (τα οποία βέβαια τις επεκτείνουν κατάλληλα για να δημιουργήσουν πιο εξειδικευμένα αντικείμενα).
- Το **GDI**, που περιέχει τις κλάσεις σχεδίασης (Graphics Device Interface). Θα δείτε ότι περιέχει επιφάνειες, πινέλα, μολύβια και χρώματα!
- Τα **Windows**, τα οποία περιέχουν τις κλάσεις που δημιουργούν τα διάφορα είδη παραθύρων.

Πολύ ενδιαφέρουσες οι παρενέργειες: Περνάμε ένα παράθυρο πάνω από την εφαρμογή μας κι αυτή μοιάζει να τρελαίνεται! Δεν είναι όμως έτσι. Διαβάστε το άρθρο και θα έχετε τη σωστή εξήγηση. Αν θέλετε να δείτε κάτι ακόμα, προσθέστε τη γραμμή self.Bind(wx.EVT_MOVE, self.OnPaint) κάτω από την πρώτη Bind και δείτε τι γίνεται όταν μετακινείτε το παράθυρο στην οθόνη!



- Το **misc**, το οποίο περιέχει διάφορες άλλες κλάσεις που δεν φαίνεται να ανήκουν σε κάποιες από τις προηγούμενες κατηγορίες.

Θα τα κατανοήσετε καλύτερα όλα αυτά φυσικά, μόλις δούμε μερικά παραδείγματα!

Εγκατάσταση wxPython

Την Python προφανώς την έχετε ήδη (κι αν όχι θα πρέπει να ανατρέξετε στα πρώτα άρθρα της σειράς μας στην Python και το PyGame), άρα χρειάζεστε μόνο την αντίστοιχη βιβλιοθήκη wxPython για το λειτουργικό σας.

Windows. Επισκεφτείτε την τοποθεσία http://wxpython.org/download.php και κατεβάστε το αντίστοιχο 32 ή 64bit Windows binary. Μπορείτε να κατεβάσετε είτε τη σταθερή (stable) είτε την υπό εξέλιξη (development) έκδοση.

OS X. Για όλες τις σύγχρονες εκδόσεις του OS X, κατεβάστε το 64bit development release από την ίδια σελίδα (http://wxpython.org/download.php). Οι σταθερές εκδόσεις λειτουργούν μόνο με 32bit Python και δεν υπάρχει λόγος να παιδευτείτε άδικα, προσπαθώντας να πείσετε το OS X να εκτελέσει την Python σε 32bit mode!

Linux. Θα βρείτε το wxPython στο repository της διανομής σας. Για παράδειγμα, στο ubuntu γράψτε sudo apt-get install python-wxgtk2.6

FreeBSD. Εγκαταστήστε το port x11-toolkits/py-wxPython28-unicode. Μη το μεταγλωττίσετε σε μέρα καύσωνα, όμως! Αν το κάνετε και το μετανιώσετε, μην πείτε ότι δεν σας προειδοποιήσαμε.

Το πρώτο σας πρόγραμμα σε wxPython

Το πρώτο μας πρόγραμμα θα υπολογίζει μια κλιμακωτή χρέωση της ΔΕΗ. Αστειευόμαστε, φυσικά! Ποτέ, μα ποτέ δεν θα σας δώσουμε τόσο βαρετά προγράμματα. Ωστόσο, αναγκαστικά το πρώτο μας πρόγραμμα θα πρέπει να είναι πολύ απλό και κατά παράδοση θα είναι το Hello World.

Χρησιμοποιήστε λοιπόν τον αγαπημένο σας editor και γράψτε το παρακάτω πρόγραμμα:

```
import wx
app = wx.App()
frame = wx.Frame(None, wx.ID_ANY, "Hello World")
caption = wx.StaticText(frame, label="Hello World of wxPython!")
frame.Show(True)
app.MainLoop()
```

Αποθηκεύστε το ως hellowx.py και εκτελέστε το γράφοντας στη γραμμή εντολών:

```
python hellowx.py
```

Τελικά δεν ήταν και τόσο δύσκολο -- και σίγουρα δεν ήταν 200 γραμμές. Το αποτέλεσμα βέβαια δεν είναι και πολύ εντυπωσιακό, αλλά τουλάχιστον είναι μια πραγματική εφαρμογή γραφικών, σε έξι μόνο γραμμές. Πώς λειτουργεί, όμως; Ας δούμε τον κώδικα, γραμμή προς γραμμή.

```
import wx
```

Προφανώς, δηλώνουμε ότι θα χρησιμοποιήσουμε τη βιβλιοθήκη wx.

```
app = wx.App()
```

Δημιουργεί ένα αντικείμενο τύπου "εφαρμογής". Κάθε εφαρμογή σε wxPython περιέχει ένα τέτοιο αντικείμενο.

```
frame = wx.Frame(None, wx.ID_ANY, "Hello World")
```

Δημιουργεί ένα αντικείμενο τύπου Frame. Μην σας μπερδεύει η αλλαγή ορολογίας: frame είναι το παράθυρό μας! Η πρώτη παράμετρος, η None, αναφέρεται στο γονικό παράθυρο, αυτό δηλαδή που περιέχει το παράθυρο μας. Προφανώς το παράθυρό μας δεν περιέχεται μέσα σε κάποιο άλλο για αυτό έχει και τιμή None. Το wx.ID_ANY αναφέρεται σε ένα αναγνωρι-

στικό αριθμό ID, ο οποίος αποδίδεται σε κάθε widget. Στις περισσότερες περιπτώσεις δεν θέλουμε (και δεν μας χρησιμεύει) να δώσουμε κάποιο συγκεκριμένο αναγνωριστικό αριθμό και χρησιμοποιούμε το wx.ID_ANY (που στην πραγματικότητα έχει τιμή -1). Το "Hello World" είναι ο τίτλος του παραθύρου.

```
caption = wx.StaticText(frame, label="Hello World of wxPython!")
```

Η γραμμή δημιουργεί ένα απλό widget, στην πραγματικότητα ένα στατικό κείμενο. Η πρώτη παράμετρος, όπως ίσως ήδη μαντέψατε, είναι το γονικό παράθυρο γι' αυτό το widget. Το κείμενο που θα δείχνει το widget το δίνουμε με τον τρόπο που βλέπετε. Στην Python, αν μια συνάρτηση έχει προαιρετικές παραμέτρους και θέλουμε να παραλείψουμε κάποιες ενδιάμεσες αλλά να δώσουμε τιμή σε μια επόμενη, χρειάζεται να αναφερθούμε σ' αυτή με το όνομά της (named arguments). Εδώ έχουμε παραλείψει το id, το οποίο είναι προαιρετικό. Θα μπορούσαμε να γράψουμε

```
caption = wx.StaticText(frame, wx.ID_ANY, "Hello World of
wxPython!")
```

αλλά παραλείποντας το id πρέπει πλέον να δώσουμε το label με το όνομά του:

```
caption = wx.StaticText(frame, label="Hello World of wxPython!")
```

Τέλος, πρέπει να δείξουμε το frame που φτιάξαμε:

```
frame.Show(True)
```

Το True (προεπιλεγμένη τιμή που θα μπορούσε πάλι να παραλειφθεί) σημαίνει ότι πράγματι θέλουμε να δείξουμε το frame. Μια τιμή False θα το εξαφάνιζε από την οθόνη! Για να εκτελεστεί η εφαρμογή, πρέπει να εκτελέσουμε τη μέθοδο MainLoop, στο αντικείμενο της εφαρμογής μας:

```
app.MainLoop()
```

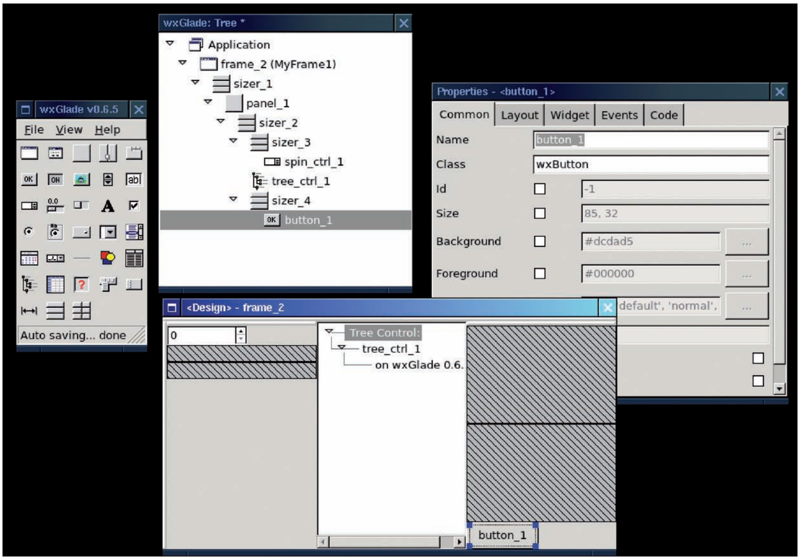
Το MainLoop() βάζει την εφαρμογή μας σε ένα βρόχο επεξεργασίας συμβάντων, όπου απλά λαμβάνει τα μηνύματα από το λειτουργικό και τα επεξεργάζεται κατάλληλα. Παράδειγμα: Ένα μήνυμα είναι το κλείσιμο της εφαρμογής, που συνήθως έρχεται όταν ο χρήστης πιέσει το (Χ) στο παράθυρο.

Η επεξεργασία των συμβάντων γίνεται με συναρτήσεις που καλούνται event handlers και οι οποίες, προφανώς, δεν γράφονται μόνες τους, ως δια μαγείας! Είναι δική μας δουλειά να γράψουμε μια σειρά από handlers για όλα τα μηνύματα στα οποία θέλουμε να αντιδρά η εφαρμογή μας. Κάποια βέβαια μπορεί να τα αγνοεί. Αν ωστόσο δοκιμάσετε να κλείσετε το παράθυρο τώρα, θα δείτε ότι γίνεται κανονικά αν και *δεν* έχουμε γράψει σχετικό κώδικα. Είναι που το μήνυμα λαμβάνεται από τις βασικές κλάσεις του wxPython, οι οποίες ξέρουν --σε αυτή την περίπτωση τουλάχιστον-- να κάνουν Το Σωστό (TM). Αν όμως αρχίσετε να κάνετε κλικ, π.χ. μέσα στο παράθυρο, δεν θα γίνει τίποτα. Γιατί πολύ απλά, το μήνυμα ναι μεν λαμβάνεται αλλά δεν υπάρχει τίποτα στις βασικές κλάσεις της wxPython που να λέει τι πρέπει να γίνεται, όταν κάνουμε κλικ σε ένα StaticText.

Hello World, απόπειρα 2

Ο τρόπος με τον οποίο έχουμε γράψει το Hello World είναι πολύ απλοϊκός: Στην πραγματικότητα, ένας πιο έμπειρος προγραμματιστής θα το έγραφε όπως βλέπετε παρακάτω.

```
import wx
class HelloWorld(wx.Frame):
  def __init__(self, *args, **kw):
    super(HelloWorld,self).__init__(*args,**kw)
    self.ShowMessage()
  def ShowMessage(self):
    caption = wx.StaticText(self, label="Hello World of
wxPython!")
    self.Show(True)
def main():
  hello = wx.App()
  HelloWorld(None, title="Hello World")
  hello.MainLoop()
if __name__ == '__main__':
  main()
```



Προγραμματισμός σε περιβάλλον γραφικών σημαίνει πολλές φορές και διάφορα βοηθητικά εργαλεία, τα οποία μας επιτρέπουν να σχεδιάζουμε παράθυρα, διαλόγους κ.λπ. και να τοποθετήσουμε πάνω τους widgets. Όλα καλά, αρκεί να μη μας γράφουν κώδικα αόρατο ή ακατάληπτο (ακούς, Visual Basic;) Το wxGlade που θα δούμε σε επόμενα άρθρα είναι ένα τέτοιο περιβάλλον, για ανάπτυξη προγραμμάτων με βάση τα wxWidgets. Περιττό ν' αναφέρουμε ότι και το ίδιο είναι γραμμένο σε Python και wxPython.

Ελπίζω να μην πάθατε σοκ, από το μέγεθος του προγράμματος (εδώ γλιτώσατε από το PyGame, τι λέμε τώρα) γιατί αν το τρέξετε θα διαπιστώσετε ότι κάνει ακριβώς το ίδιο πράγμα με το προηγούμενο! Τότε γιατί να μπούμε σε όλον αυτόν το μπελά; Μην ξεχνάτε ότι για να εκμεταλλευτούμε πλήρως τον αντικειμενοστρεφή προγραμματισμό πρέπει να δημιουργούμε τα δικά μας classes, όπως κάναμε και στο PyGame. Τυπικά, δεν θα χρησιμοποιούμε απευθείας τα classes που παρέχει η wxPython αλλά θα φτιάχνουμε δικά μας που θα βασίζονται σε αυτά (derived classes, όπως λέγονται), και στα οποία θα έχουμε τη δυνατότητα να προσθέτουμε τις δικές μας λειτουργίες παρακάμπτοντας ή επεκτείνοντας αυτές που παρέχει η βασική κλάση. Στο προηγούμενο παράδειγμα φτιάξαμε δική μας κλάση για το Frame που θα χρησιμοποιήσουμε. Αυτή κληρονομεί τα πάντα από την βασική κλάση wx.Frame, στην οποία βασίζεται:

```
class HelloWorld(wx.Frame):
```

Όμως όπως παρατηρείτε, έχουμε επεκτείνει τον constructor (__init__) ώστε όταν δημιουργείται το παράθυρο (κι αφού κληθεί ο constructor της γονικής κλάσης) να καλεί και το member function ονόματι ShowMessage, το οποίο θα δείξει το μήνυμά μας. Με λίγα λόγια, έχουμε ένα ειδικό παράθυρο “Hello World”. Κάθε φορά που δημιουργείται, περιέχει και το μήνυμά μας μέσα. Εμφανίζει μάλιστα και τον εαυτό του, αυτόματα (self.Show(True)).

Η παρακάτω γραμμή είναι λίγο μυστηριώδης:

```
def __init__(self, *args, **kw):
```

Θεωρήστε απλά ότι είναι ο τρόπος για να περάσουμε οποιεσδήποτε παραμέτρους στον constructor της βασικής κλάσης, που καλούμε αμέσως μετά:

```
super(HelloWorld,self).__init__(*args,**kw)
```

Παρατηρήστε ότι κατά τη δημιουργία του αντικειμένου HelloWorld, έχουμε δύο παραμέτρους:

```
HelloWorld(None, title="Hello World")
```

όπου φυσικά None είναι το γονικό παράθυρο και title ο τίτλος του παραθύρου μας (παραλείψαμε το ID ενδιάμεσα). Δεν χρειάζεται ν’ ανησυχείτε για την ώρα για τα *args και **kw. (Αλλά σε περίπτωση που αυτό δεν σας αφήσει να κοιμηθείτε το βράδυ, διαβάστε εδώ: http://pythonfilter.com/blog/what-do-args-kw-kwargs-mean.html). Κι ας πάμε τώρα σε ένα λίγο πιο όμορφο και πολύπλοκο προγραμματάκι...

Drawing in wxPython: Marquee

Το παρακάτω πρόγραμμα μοιάζει λίγο με το Marquee του TI-99 (http://bit.ly/ti99marquee). Εκτός του ότι δεν έχει ήχο, έχει κάποιες ενδιαφέρουσες, εχμ, παρενέργειες!

```
import wx
from random import randint
class DrawPanel(wx.Frame):
  def __init__(self):
    super(DrawPanel,self).__init__(parent=None,
size=(640,480),title="Drawing Example")
    self.Bind(wx.EVT_PAINT, self.OnPaint)
  def OnPaint(self, e):
    dc = wx.PaintDC(self)
    dc.Clear()
    for i in range(0,640,32):
       dc.SetPen(wx.Pen(wx.Colour(randint(0,255),randint(0,255),
randint(0,255)), 32))
          dc.DrawLine(i, 0, i, 480)
def main():
  app = wx.App(False)
  frame = DrawPanel()
```

```
    frame.Show()
    app.MainLoop()
if __name__ == '__main__':
    main()
```

Εκτελέστε το και θα δείτε κάποιες χρωματιστές μπάρες τυχαίου χρώματος (randint is your friend) στο παράθυρό σας. Για δοκιμάστε όμως να μετακινήσετε ένα άλλο παράθυρο πάνω στο Marquee... Γιατί συμβαίνει αυτό; Κάθε φορά που το παράθυρο μας έχει "καταστραφεί" από κάποιο άλλο (ή έχει μόλις δημιουργηθεί επειδή τρέξαμε την εφαρμογή), το λειτουργικό στέλνει ένα μήνυμα wx.EVT_PAINT το οποίο, πρακτικά, σημαίνει: "Ξέρεις, κάτι τρέχει με το παράθυρό σου, κάποιος το κάλυψε, αυτά που είχε μέσα διαλύθηκαν. Ξανασχεδίασε τα, σε παρακαλώ."

Είναι λίγο σοκαριστικό: Το πρόγραμμά μας πρέπει να είναι πάντα έτοιμο να ξανασχεδιάσει τα περιεχόμενα του παραθύρου του, είτε πρόκειται για μια μικρή γωνία είτε για όλο το παράθυρο. Πώς όμως αντιλαμβανόμαστε εμείς το event; Κοιτάξτε λίγο τον constructor:

```
self.Bind(wx.EVT_PAINT, self.OnPaint)
```

Όταν λάβουμε μήνυμα wx.EVT_PAINT, εκτελείται το member function OnPaint. Το dc, device context, είναι ένα αντικείμενο που αντιπροσωπεύει την περιοχή του παραθύρου που χρειάζεται να σχεδιαστεί ξανά: Αυτή που επικαλύφθηκε ή και όλο το παράθυρο, αν μόλις τώρα τρέξαμε την εφαρμογή! Είμαστε σίγουροι ότι μπορείτε να φανταστείτε πώς γίνεται η σχεδίαση: Δημιουργούμε ένα μολύβι (pen) τυχαίου χρώματος και πάχους 32 pixels και το επιλέγουμε για χρήση στο device context (όλα σε ένα):

```
dc.SetPen(wx.Pen(wx.Colour(randint(0,255),randint(0,255),
randint(0,255)), 32))
```

Σχεδιάζουμε τη γραμμή:

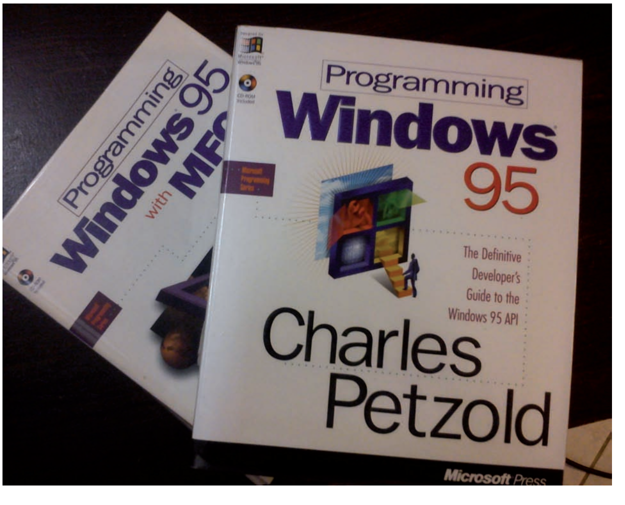
```
dc.DrawLine(i, 0, i, 480)
```

'Οπως όμως αναφέραμε παραπάνω, ο λόγος της περίεργης συμπεριφοράς του προγράμματός μας είναι: το dc περιέχει *μόνο* την περιοχή που χρειάζεται να σχεδιαστεί! Άρα, το (640,480) που επιχειρούμε με το loop έχει νόημα μόνον όταν πρόκειται να σχεδιαστεί όλο το παράθυρο από την αρχή! Και ναι, για όλους εσάς που γράφατε Visual Basic, την επανασχεδίαση του παραθύρου την έκανε η γλώσσα για εσάς. Είδατε τι ωραία πράγματα χάνατε; :D

Μην περιμένετε μέχρι το επόμενο άρθρο μας για να πειραματιστείτε με το wxPython: Έχετε άφθονο διάβασμα στο http://wiki.wxpython.org

Για άλλη μια φορά, καλά ξενύχτια!

Εντάξει, θα το ομολογήσω: Οι πρώτες εφαρμογές γραφικών που έγραψα ήταν για Windows κι αυτά τα δύο βιβλία ήταν πολύτιμα. Το «Programming Windows, Charles Petzold» αντιμετωπίζει τον προγραμματισμό Windows σε C χρησιμοποιώντας το API με τον κλασικό τρόπο. Το «Programming Windows with MFC» από την άλλη, χρησιμοποιεί τα Microsoft Foundation Classes και την αντικειμενοστρεφή C++. Όποιος σας καυχιέται ότι είναι Windows Programmer και δεν έχει διαβάσει τον Petzold, μπορείτε άνετα να τον κοροϊδεύετε. Θα σας βοηθήσουμε και εμείς αν χρειαστεί :) Αν δείτε το Hello World σε MFC, θα εντυπωσιαστείτε με το πόσο μοιάζει με τα δικά μας αρχικά προγράμματα!



Skill: Intermediate

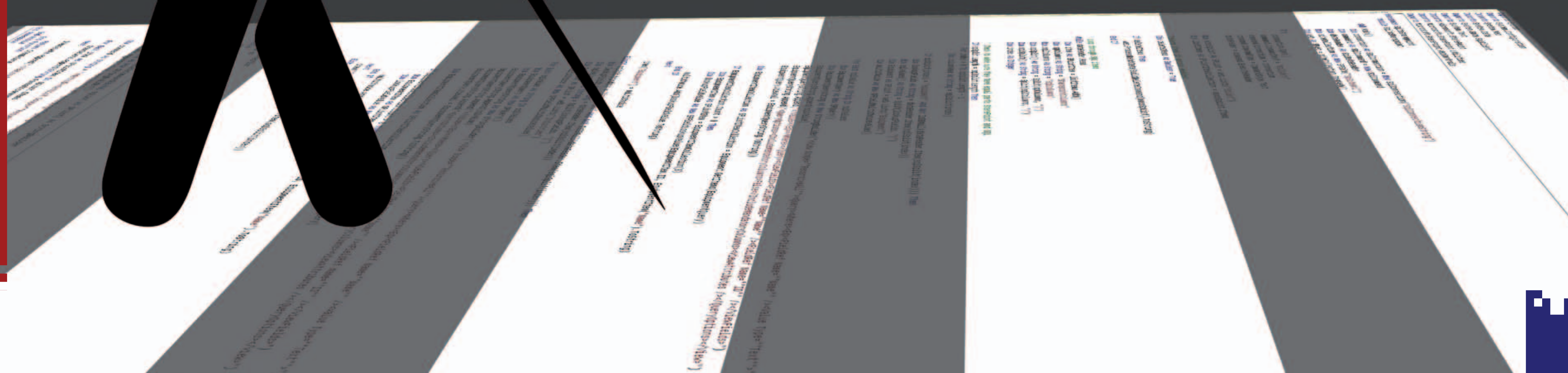
Tags: input sanitization, injection point, blind SQL injection, LAMP, themole

ΕΠΙΘΕΣΗ ΣΤΑ ΤΥΦΛΑ!

(Μέρος 1ο)

Δεν τρελαθήκαμε, ούτε βέβαια σκοπεύουμε να επιτεθούμε σε όλο το Internet. Θα σας δείξουμε μόνο ένα παιχνίδι ερωτήσεων, το οποίο παίζεται με ευπαθείς δικτυακές εφαρμογές. Μ’ αυτό το παιχνίδι, χωρίς να ζητάμε και χωρίς να βλέπουμε αυτό που πραγματικά μας ενδιαφέρει, θα εξαγάγουμε κρίσιμες πληροφορίες... από τα συμφραζόμενα.

by 41i3n

Ίσως ήδη παρεξηγήσατε τον τίτλο του άρθρου, αν κι εδώ που τα λέμε είναι αρκετά παραπλανητικός! Το σίγουρο πάντως είναι ότι δεν εννοούμε μια επίθεση με κλειστά μάτια, π.χ., σε οποιονδήποτε ή οτιδήποτε βρεθεί μπροστά μας και με απρόβλεπτες συνέπειες. Αντίθετα, εννοούμε μια επίθεση η οποία βασίζεται σε όχι και τόσο φανερά δεδομένα. Πιο σωστά, σε δεδομένα που κρύβονται πίσω από γεγονότα, τα οποία συνήθως θεωρούμε αδιάφορα. Μιλάμε για το λεγόμενο blind SQL injection. Πρόκειται για μια επίθεση που βρίσκεται στις πρώτες θέσεις του OWASP Top 10, εδώ και πολλά χρόνια (`www.owasp.org/index.php/Top_10_2010-A1`). Μάλιστα, αξίζει να πούμε ότι ενώ χαρακτηρίζεται από μέτριο βαθμό δυσκολίας, έχει σοβαρότατες επιπτώσεις!

Η απαραίτητη θεωρία

Υπάρχουν χιλιάδες web applications Εκεί Έξω (TM), οι οποίες μας εξυπηρετούν με τον ένα ή τον άλλο τρόπο. Οι περισσότερες από αυτές βασίζονται σε κάποιο σύστημα διαχείρισης περιεχομένου (Content Management System) και, συνεπώς, τις περισσότερες φορές τα δεδομένα βρίσκονται αποθηκευμένα σε κάποια βάση. Αυτά τα δεδομένα ανασύρονται από τη βάση και σερβίρονται στον τελικό χρήστη --στον επισκέπτη της εκάστοτε web app--, από προγράμματα γραμμένα σε κάποια scripting language (http://en.wikipedia.org/wiki/Server-side_scripting). Ουσιαστικά, τα εν λόγω προγράμματα αναλαμβάνουν να μεταφράσουν τις επιλογές του χρήστη σε ένα κατάλληλο ερώτημα (query) προς τη βάση δεδομένων, χρησιμοποιώντας τη γλώσσα SQL (Structured Query Language). Για παράδειγμα, ας δούμε ένα τυπικό HTTP request, με μια παράμετρο ονόματι id που μεταφέρεται στον server με τη μέθοδο GET:

```
http://domain.tld/app/itemView?id=1
```

Αυτό το αίτημα, το οποίο ενδέχεται να προέκυψε από το κλικ του χρήστη σε κάποιο σύνδεσμο ή σε κάποιο κουμπάκι κάποιας φόρμας, θα μεταφραστεί σε query προς τη βάση δεδομένων και θα αποκτήσει μια τέτοια μορφή:

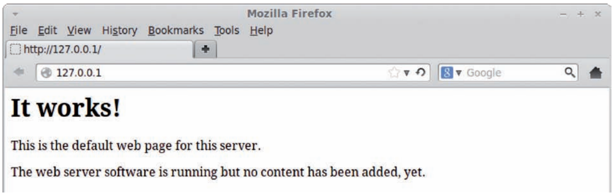
```
SELECT * FROM items WHERE id=1
```

Τα αποτελέσματα του ερωτήματος προς τη βάση δεδομένων θα επιστραφούν στο χρήστη, μέσω του HTTP response. Φυσικά, ο χρήστης δεν θα παραλάβει μια ξερή απάντηση. Διάφορα server-side scripts θα φροντίσουν να μορφοποιήσουν και να ντύσουν κατάλληλα τα δεδομένα, μέσα σε κώδικα HTML. Προφανώς, η όλη διαδικασία είναι διάφανη ως προς τον χρήστη, ο οποίος δεν βλέπει ούτε τα queries προς τη βάση, ούτε τα scripts που μορφοποιούν την απάντηση της βάσης.

Τις περισσότερες φορές, οτιδήποτε προέρχεται από το χρήστη --ή μπορεί να τροποποιηθεί από αυτόν--, υποβάλλεται σε μια σειρά ελέγχων (input sanitization). Αυτό συμβαίνει ώστε κάθε προσπάθεια εκτέλεσης ανεπιθύμητων ερωτημάτων[1] προς τη βάση να καταλήγει σ’ ένα graceful error message, με την ελάχιστη διαρροή πληροφοριών (minimal information disclosure). Όμως τι γίνεται στις περιπτώσεις που ο έλεγχος είναι ανεπαρκής και η εκτέλεση ενός τέτοιου query καθίσταται δυνατή; Τότε, λέμε ότι υπάρχει ένα SQL injection point. Πρόκειται για ένα κενό ασφαλείας, δηλαδή, το οποίο επιτρέπει την αυθαίρετη εκτέλεση ερωτημάτων προς

1. Τα ερωτήματα σχηματίζονται στη γλώσσα SQL και όλες οι εκδοχές της συγκεκριμένης γλώσσας επιτρέπουν πολύ περισσότερες ενέργειες από μία απλή αναζήτηση. Έτσι, μερικά επικίνδυνα ερωτήματα προς τη βάση θα μπορούσαν να είναι τα εξής: Ποιος είναι ο κωδικός του root; Σβήσε τον τάδε πίνακα. Γράψε το τάδε string στον τάδε πίνακα κ.ο.κ.

τη βάση κι έχει ως συνέπεια την κλοπή κρυφών ή/και ευαίσθητων πληροφοριών. Οι σχετικές επιθέσεις ονομάζονται SQL injections κι ένα παρακλάδι τους --αυτό που θα μελετήσουμε--, ονομάζεται blind SQL injections.



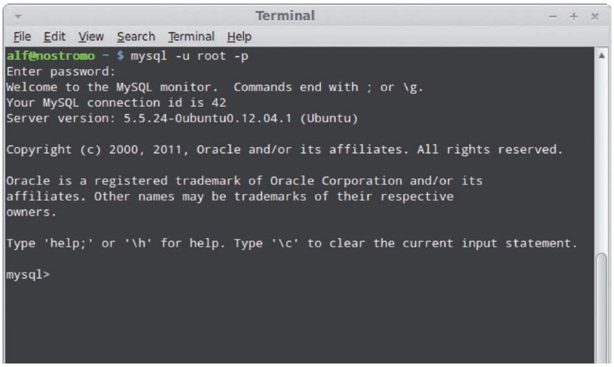
Αφού εγκαταστήσουμε το πακέτο LAMP, μπορούμε να τσεκάρουμε τη λειτουργία του Apache web server με τη βοήθεια ενός οποιουδήποτε web browser. Όπως φαίνεται στην εικόνα, ο Apache είναι πανέτοιμος και δέχεται συνδέσεις κανονικά.

Συνήθως, το input sanitization εμποδίζει την εκτέλεση κραυγαλέων ερωτημάτων προς τη βάση και τελικά αποτρέπει την άμεση αποκάλυψη κρυφών πληροφοριών. Ωστόσο, ενδέχεται να επιτρέπονται ορισμένα είδη ερωτημάτων, τα οποία είναι φαινομενικά αθώα αλλά μπορούν να οδηγήσουν στην έμμεση αποκάλυψη μυστικών πληροφοριών. Σκεφτείτε την περίπτωση που η εκτέλεση απαγορευμένων ερωτημάτων --ή ερωτημάτων με αποτέλεσμα το λογικό FALSE-- οδηγεί στην εμφάνιση ενός μηνύματος λάθους, ενώ η εκτέλεση ενός επιτρεπόμενου ερωτήματος --ή ενός ερωτήματος με αποτέλεσμα το λογικό TRUE-- να οδηγεί στην εμφάνιση μιας κανονικής σελίδας. Αυτή η συμπεριφορά μοιάζει αρκετά ασφαλής για τη δικτυακή εφαρμογή και για τη βάση δεδομένων, αλλά δεν είναι! Με αυτή τη συμπεριφορά που παρουσιάζουν τα server-side scripts είναι αδύνατον να τροποποιήσουμε τη βάση ή να ζητήσουμε τον κωδικό κάποιου χρήστη και να τον δούμε. Με ένα ερώτημα του τύπου «δείξε μου το στοιχείο password, του πίνακα users και της γραμμής που περιέχει το admin», θα παίρναμε ένα ωραιότατο μήνυμα λάθους και θα ησυχάζαμε. Ωστόσο, αυτή η συμπεριφορά των scripts επιτρέπει ένα παιχνίδι ερωτήσεων με τον server, στις οποίες θα παίρνουμε "μονολεκτική" απάντηση: Είτε ένα "ναι" (κανονική σελίδα περιεχομένου) είτε ένα "όχι" (σελίδα με μήνυμα λάθους). Για να φανεί η επικινδυνότητα αυτού του παιχνιδιού, σκεφτείτε ένα ερώτημα της μορφής «δείξε μου το περιεχόμενο με id=1 AND ο πρώτος χαρακτήρας του admin password είναι 'A'». Σε αυτό το ερώτημα έχουμε συμπεριλάβει τη λογική πράξη ΚΑΙ (AND). Το πρώτο σκέλος του ερωτήματος ζητάει την πρόσβαση σε νόμιμο περιεχόμενο, ενώ το δεύτερο υπονοεί έναν έλεγχο (είναι όντως 'A' το πρώτο γράμμα του password του admin;). Αν με την εκτέλεση αυτού του ερωτήματος εμφανιστεί μια κανονική σελίδα περιεχομένου, τότε η λογική πράξη ΚΑΙ είχε αποτέλεσμα TRUE. Αυτό σημαίνει ότι το πρώτο γράμμα του password του admin ήταν *όντως* το Α! Αντίθετα, αν η εκτέλεση του παραπάνω ερωτήματος εμφανίσει ένα μήνυμα λάθους, συμπεραίνουμε ότι η λογική πράξη ΚΑΙ είχε σαν αποτέλεσμα το FALSE. Επομένως, γνωρίζουμε πλέον ότι ο κωδικός του admin δεν ξεκινά με το Α και μπορούμε να προχωρήσουμε στο επόμενο γράμμα... ;)

Εφόσον ο Apache λειτουργεί, μπορούμε να προχωρήσουμε στη ρύθμιση της MySQL. Συγκεκριμένα, πρέπει να προετοιμάσουμε τη βάση και τους αντίστοιχους πίνακες, τους οποίους θα χρησιμοποιεί το ευπαθές web application. Για το σκοπό αυτό, θα συνδεθούμε στη MySQL ως root...

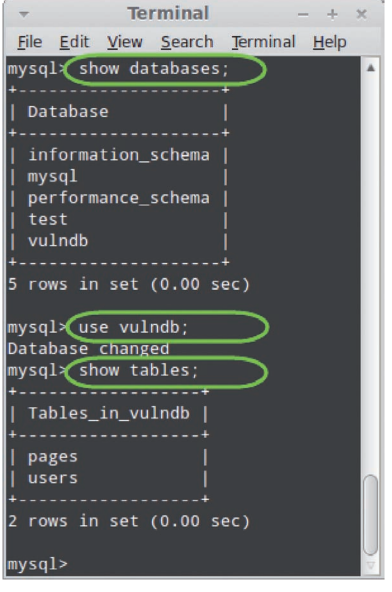
Προετοιμασία

Είναι γεγονός ότι υπάρχουν πολλές βάσεις δεδομένων (MS SQL, MySQL, PostgreSQL κ.ά.), πολλές γλώσσες scripting (PHP, ASP, JSP, CGI κ.λπ.), πολλές μέθοδοι για input sanitization και πολλές ακόμα τεχνικές ασφάλειες (Web Application Firewalls, Intrusion Detection Systems κ.ο.κ.). Αυτή η ποικιλία καθιστά κάθε επίθεση SQL injection ξεχωριστή κι αρκετά πε-



ρίπλοκη. Στη δική μας περίπτωση θα απλουστεύσουμε λίγο τα πράγματα, αφού ο στόχος μας είναι η εξοικείωση με τη λογική της επίθεσης κι όχι η αντιμετώπιση όλων των μηχανισμών άμυνας ενός server, ούτε η ανάλυση κάθε πιθανού σεναρίου. Για την παρουσίασή μας, λοιπόν, θα σηκώσουμε ένα απλοϊκό web application σε ένα δικό μας μηχάνημα (Repeat after me: Δεν κάνουμε επίθεση σε ξένα μηχανήματα χωρίς την άδεια του ιδιοκτήτη, ούτε καν για εκπαιδευτικούς λόγους.) Συγκεκριμένα, θα εγκαταστήσουμε το περιβάλλον LAMP[2] σε ένα σύστημα με το Linux Mint 13 Maya – MATE Edition. Δεν υπάρχει κάποιος ιδιαίτερος λόγος για την επιλογή αυτή, εκτός του ότι θέλαμε να δοκιμάσουμε το MATE. Φυσικά, εσείς μπορείτε να χρησιμοποιήσετε οποιαδήποτε άλλη διανομή ή ακόμα και τα Windows. Παρεμπιπτόντως, σε αυτή την περίπτωση θα σας φανούν πολύτιμα τα σχετικά άρθρα του Praeto, στο site του περιοδικού: `http://deltahacker.gr/?p=89`, `http://deltahacker.gr/?p=755` και `http://deltahacker.gr/?p=809`.

Σε γενικές γραμμές, το μηχάνημα-στόχος θα χρησιμοποιεί τη βάση MySQL κι ένα πολύ απλό script σε PHP, το οποίο θα κάνει ελάχιστο έως ανύπαρκτο sanitization. Για την πραγματοποίηση των επιθέσεων θα αξιοποιήσουμε μερικά δικά μας scripts (γραμμένα σε Perl), καθώς κι ένα έτοιμο εργαλείο για SQL injection, με το όνομα themole3. Στη συνέχεια θα εξετάσουμε από κοντά όλη την προετοιμασία του τρωτού server. Σημειώστε ότι όλα τα script που θα δούμε βρίσκονται στο ακόλουθο πακέτο: `http://bit.ly/dh012BlindSQLi` (προσοχή στα πεζοκεφαλαία).



Αφού φορτωθεί το αρχείο με τα έτοιμα SQL statements, είμαστε σε θέση να τσεκάρουμε ότι δημιουργήθηκε η βάση (vulndb) που θα χρησιμοποιεί η ευάλωτη εφαρμογή μας. Μπορούμε να επεκτείνουμε τους ελέγχους, ρίχνοντας μια ματιά στα περιεχόμενα της νέας βάσης.

Το υπόστρωμα

Ξεκινάμε κάνοντας ένα πλήρες update στα repositories του Mint κι εγκαθιστώντας στη συνέχεια το πακέτο LAMP:

```
alf@nostromo ~ $ sudo apt-get update
```

```
alf@nostromo ~ $ sudo apt-get install lamp-server
```

Κατά τη διάρκεια της εγκατάστασης θα ζητηθεί ένα password για τον χρήστη root -- της MySQL. Δίνουμε ένα ισχυρό συνθηματικό και φροντίζουμε να το θυμόμαστε, γιατί θα μας χρειαστεί πολύ σύντομα. Με το τέλος της εγκατάστασης μπορούμε να δοκιμάσουμε τον Apache web server δίνοντας στον Firefox του Mint τη διεύθυνση `http://127.0.0.1`. Εφόσον πήγαν όλα καλά, θα μας καλωσορίσει το γνωστό μήνυμα «It works!". Αμέσως μετά, σειρά έχει η σύνδεση στη MySQL, ως root (της MySQL). Σκοπός μας τώρα είναι η δημιουργία της βάσης την οποία και θα χρησιμοποιεί το τρωτό μας web application. Για τη σύνδεση στη MySQL σαν root, δίνουμε το ακόλουθο:

```
alf@nostromo ~ $ mysql -u root -p
```

2. Linux – Apache – MySQL – PHP
3. `http://themole.nasel.com.ar/`

Μόλις δώσουμε το password του root για τη MySQL, το σύστημα θα μας καλωσορίσει και θα βρεθούμε στη γραμμή εντολών της MySQL. Για τη δημιουργία της βάσης και των πινάκων που θα χρειαστεί το web app, αρκεί να φορτώσουμε ένα αρχείο με μερικά SQL statements. Σε αυτό το στάδιο, αν δεν το έχετε κάνει ήδη, πρέπει να κατεβάσετε το πακέτο που αναφέραμε νωρίτερα (αυτό από το http://bit.ly/dh012BlindSQLi). Το αρχείο που μας ενδιαφέρει είναι το create_db.sql και για τη φόρτωσή του αρκεί να δώσουμε κάτι τέτοιο:

```
mysql> source Desktop/create_db.sql
```

Προφανώς, στη θέση του Desktop θα πρέπει να βάλετε τον κατάλογο στον οποίο έχετε αποσυμπιέσει το πακέτο ZIP. Επεξηγηματικά, να πούμε ότι τα statements που περιλαμβάνει το create_db.sql θα δημιουργήσουνε μια βάση με το όνομα vulndb και τους πίνακες users και pages. Ο πίνακας users έχει δύο στήλες (username και password), ενώ ο πίνακας pages έχει τρεις στήλες (id, page_content και author). Επιπρόσθετα, εισάγονται μερικοί χρήστες και τα αντίστοιχα password, σε κρυπτογραφημένη μορφή, καθώς επίσης και περιεχόμενο στον πίνακα pages. Τέλος, δημιουργείται ο χρήστης dbo με password το from_outer_space. Ο εν λόγω χρήστης έχει πλήρη δικαιώματα στη βάση vuldb. Αυτός ο λογαριασμός θα χρησιμοποιείται αργότερα από το script μας, για τη διαχείριση της βάσης. Αναρωτιέστε γιατί δεν χρησιμοποιούμε το χρήστη root γι' αυτή τη δουλειά; Είπαμε να κάνουμε ένα τρωτό server, αλλά αυτό δεν σημαίνει ότι θα τα ισοπεδώσουμε όλα! Ο root της MySQL έχει πρόσβαση σε *όλες* τις βάσεις κι αν με κάποιο τρόπο αποκαλυφθούν τα δικά του credentials, μπορούμε να ξεχάσουμε *όλες* τις βάσεις του συστήματός μας. Επομένως, η δημιουργία ενός ξεχωριστού χρήστη για τη διαχείριση κάθε βάσης αποτελεί ζήτημα υψίστης ασφάλειας. Τώρα όμως ξεφύγαμε από το θέμα μας... Εφόσον λοιπόν φορτώθηκαν κι εκτελέστηκαν τα SQL statements, μπορούμε να κάνουμε μερικές δοκιμές. Ουσιαστικά, μπορούμε να ζητήσουμε από τη MySQL να προβάλει τη βάση και τους πίνακές της, για να βεβαιωθούμε ότι έγιναν όλα σωστά. Για παράδειγμα, για να δούμε όλες τις βάσεις δεδομένων και να ελέγξουμε αν δημιουργήθηκε η vulndb, αρκεί να δώσουμε κάτι τέτοιο:

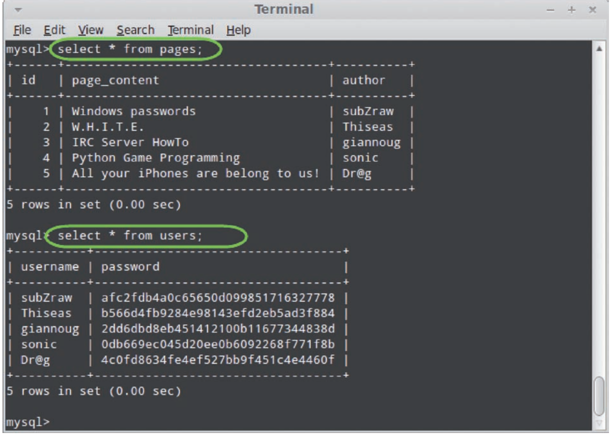
Οι έλεγχοι στη βάση vulndb μπορούν να συνεχιστούν μέχρις εσχάτων. Η συγκεκριμένη βάση περιλαμβάνει όλο το περιεχόμενο που θα σερβίρει αργότερα η ευάλωτη δικτυακή εφαρμογή...

```
mysql> show databases;
```

Στη συνέχεια, για να δούμε τους πίνακες της vulndb αρκεί να δώσουμε το εξής:

```
mysql> use vulndb;
Database changed
mysql> show tables;
```

Αφού επιβεβαιώσουμε ότι είναι όλα καλά, μπορούμε να επιστρέψουμε στη γραμμή εντολών του BASH, δίνοντας ένα exit.



```
Terminal
File Edit View Search Terminal Help
mysql> select * from pages;
+------+----------------------------------+----------+
| id   | page_content                     | author   |
+------+----------------------------------+----------+
|    1 | Windows passwords                | subZraw  |
|    2 | W.H.I.T.E.                       | Thiseas  |
|    3 | IRC Server HowTo                 | giannoug |
|    4 | Python Game Programming          | sonic    |
|    5 | All your iPhones are belong to us! | Dr@g   |
+------+----------------------------------+----------+
5 rows in set (0.00 sec)

mysql> select * from users;
+----------+----------------------------------+
| username | password                         |
+----------+----------------------------------+
| subZraw  | afc2fdb4a0c65650d099851716327778 |
| Thiseas  | b566d4fb9284e98143efd2eb5ad3f884 |
| giannoug | 2dd6dbd8eb451412100b11677344838d |
| sonic    | 0db669ec045d20ee0b6092268f771f8b |
| Dr@g     | 4c0fd8634fe4ef527bb9f451c4e4460f |
+----------+----------------------------------+
5 rows in set (0.00 sec)

mysql>
```

Η τρωτή εφαρμογή

Επόμενο σταθμό αποτελεί η δημιουργία της «τρύπιας» δικτυακής εφαρμογής. Πρόκειται για ένα υποτιθέμενο δικτυακό αρχείο (archive), με τα άρθρα του περιοδικού. Ουσιαστικά, όλη η εφαρμογή βρίσκεται στο αρχείο webapp.php, μέσα στο πακέτο BlindSQL.zip. Για την εγκατάσταση (!) της

Ο κώδικας για τη δικτυακή μας εφαρμογή. Το μαρκαρισμένο τμήμα δείχνει την περιοχή στην οποία συντελείται το έγκλημα: Το input sanitization είναι της πλάκας!

```
Terminal
File  Edit  View  Search  Terminal  Help
  GNU nano 2.2.6                File: /var/www/webapp.php

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Blind SQL Injection example</title>
</head>
<body>
<h1>Welcome to the deltaHacker's article archive!</h1>
<a href="?id=1">Article 1</a>
   
<a href="?id=2">Article 2</a>
   
<a href="?id=3">Article 3</a>
   
<a href="?id=4">Article 4</a>
   
<a href="?id=5">Article 5</a>
<p>
<?php
$db=mysql_connect("localhost","dbo","from_outer_space");
if (!$db) {
        echo "Could not connect to database: '" . mysql_error() . "' <br />\n";
} else {
        if (mysql_select_db("vulndb", $db)) {
                if ($_GET['id'] != null) {
                        $id = $_GET['id'];
                } else {
                        $id = 1;
                }
                $sql = "SELECT id,page_content,author FROM pages WHERE id=$id";
                $result = mysql_query($sql);
                $row = mysql_fetch_assoc($result);
                if ($row) {
                        echo "Article: ".$row['id'] . "<br /><br />\n";
                        echo $row['page_content'] . "<br />\n";
                        echo "by ".$row['author'] . "<br />\n";
                } else {
                        echo "Page not found.<br />\n";
                }
        } else {
                echo "Could not open to database: '" . mysql_error() . "' <br />\n";
        }
        mysql_close($db);
}
?>
</p>
</body>
</html>

^G Get Help   ^O WriteOut   ^R Read File   ^Y Prev Page   ^K Cut Text     ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is    ^V Next Page   ^U UnCut Text   ^T To Spell
```

εφαρμογής αρκεί η αντιγραφή του εν λόγω script μέσα στο Document Root του Apache. (Στη δική μας περίπτωση ήταν ο κατάλογος /var/www.) Με μια ματιά στο περιεχόμενο του webapp.php, πιστεύουμε ότι θα εντοπίσετε αμέσως το πρόβλημα...

```
if (mysql_select_db("vulndb", $db)) {
        if ($_GET['id'] != null) {
                $id = $_GET['id'];
        } else {
                $id = 1;
        }
```

Το μόνο input sanitization που γίνεται στην παράμετρο id είναι ένας έλεγχος για να διαπιστωθεί ότι δεν είναι κενή (null). Εδώ κι αν μιλάμε για ανεπαρκή έλεγχο! Θα μπορούσε τουλάχιστον να ακολουθεί μια δήλωση της μεταβλητής id, με την ακόλουθη μορφή:

```
$id = intval($_GET['id'])
```

Με αυτόν τον τρόπο, οτιδήποτε έδινε ο χρήστης θα μετατρεπόταν αυτόματα σε integer. Έτσι, θα αποκλειόταν μια μεγάλη μερίδα επικίνδυνων queries.

Στη συνέχεια δίνουμε ένα sudo restart apache2, έτσι, για τα καλορίζικα. Τώρα, αν έχουν γίνει όλα σωστά, θα μπορούμε να επισκεφθούμε το νέο μας web application, από τον αγαπημένο μας browser. Μπορούμε να κάνουμε τις δοκιμές και να παίξουμε με την εφαρμογή, επισκεπτόμενοι τη διεύθυνση `http://localhost/webapp.php`. Παρεμπιπτόντως, το συγκεκριμένο web app θα μπορούσε να χρησιμοποιηθεί και σε μελλοντικά άρθρα, σχετικά με web attacks. Τι θα λέγατε για μια γνωριμία με τα union select injections, cross site scripting, local file inclusion και directory traversal; Ας μην προτρέχουμε όμως...



Μη μας πείτε ότι δεν είναι πανέμορφο το νέο μας web application :)

Ολοκλήρωση

Όπως είπαμε παραπάνω, για την επίθεση θα χρησιμοποιήσουμε μερικά δικά μας scripts γραμμένα σε Perl, καθώς και το themole. Επομένως θα χρειαστούμε τον interpreter της Perl, καθώς επίσης και ορισμένα modules για την Python 3. Στο δικό μας περιβάλλον η Perl ήταν ήδη εγκατεστημένη, κι εδώ που τα λέμε θα τη βρείτε προεγκατεστημένη στις περισσότερες διανομές Linux. Ιδιαίτερη προσοχή θα χρειαστεί για το themole. Αρχικά πρέπει να επισκεφθούμε την επίσημη ιστοσελίδα στο http://themole.nasel.com.ar και να κατεβάσουμε το πακέτο deb. Στη συνέχεια θα το εγκαταστήσουμε δίνοντας κάτι τέτοιο:

```
alf@nostromo ~/Desktop $ sudo dpkg -i themole_0.3-1_all.deb
```

Σημειώστε ότι οφείλουμε να ακολουθήσουμε όλες τις υποδείξεις που ενδέχεται να εμφανιστούν για τα missing dependencies. Αν χρησιμοποιείτε κάποια διανομή που δεν βασίζεται στο Debian, ακολουθήστε τις οδηγίες που παρέχονται στο site του themole (`http://themole.nasel.com.ar/?q=documentation/installation`).

Τώρα, αν κουραστήκατε με την (απαραίτητη) προετοιμασία, μην ανησυχείτε! Θα σας ανταμείψουμε στο επόμενο άρθρο, στο οποίο κάτι μας λέει ότι δεν θα βαρεθείτε καθόλου και, ευτυχώς για όλους μας, ξεκινά από τη σελίδα 76 του deltaHacker που κρατάτε στα χέρια σας. Σ' αυτό, ξεκινάμε με μερικές διερευνητικές επιθέσεις, με σκοπό ν' ανακαλύψουμε το injection point, ενώ αμέσως μετά αρχίζουμε με κανονικότατες επιθέσεις blind SQL injection. Έτσι, θα δείτε την κατασκευή κατάλληλων SQL queries τα οποία καταφέρνουν να εξάγουν διάφορες κρυφές πληροφορίες -- ύπουλα κι αποτελεσματικά! Στο τέλος του άρθρου εξετάζουμε και μερικά χρήσιμα εργαλεία, με τη βοήθεια των οποίων οι σχετικές επιθέσεις απλοποιούνται σε μεγάλο βαθμό. Αλλά κάτι μας λέει ότι αυτή την τελευταία πρόταση δεν τη διαβάσατε, γιατί ήδη έχετε γυρίσει στη σελίδα 76. Ή μήπως όχι; Παράξενο πράγμα πάντως, ε; Αν το φιλοσοφήσετε λίγο, θα δείτε ότι αυτές οι τελευταίες προτάσεις είναι οιωνεί ανύπαρκτες. Χμ.

Skill: Advanced

Tags: BCM, pointer tricks, loop unroll, ASM inline

Arduino
the way of the hacker!!

(Μέρος 7/k*)

Ο έλεγχος ενός RGB LED matrix αποτελεί μια προκλητικά σύνθετη εργασία και δεν είναι τυχαίο ότι έλκει παλιούς και νέους στα ηλεκτρονικά. Πρόκειται για μια εργασία που απαιτεί καθαρή σκέψη και σχετικά υψηλό επίπεδο προγραμματιστικών γνώσεων. Εξάλλου, η χαμηλή ανάλυση των matrix παραπέμπει αυτόματα στα παιχνίδια των παλιών υπολογιστών. Σ' αυτά που όλοι έχουμε δει και όλοι θα θέλαμε να φτιάξουμε κάποια στιγμή...

by Spir@lEvolution

* όπου k θετικός ακέραιος με τουλάχιστον μια ενδιαφέρουσα ιδιότητα

Σε προηγούμενο άρθρο της σειράς μας, και συγκεκριμένα στο deltaHacker 007, εξετάσαμε το πώς υλοποιείται το Pulse Width Modulation σε software. Υπάρχουν βλέπετε περιπτώσεις στις οποίες τα κανάλια PWM που προσφέρει ο επεξεργαστής του Arduino δεν επαρκούν και τότε η στροφή στο software αποτελεί μονόδρομο. Ωστόσο, όταν υλοποιούμε ένα σύστημα PWM μεγάλης ακρίβειας (πολλών βημάτων ρύθμισης) η απόδοση του κώδικα μειώνεται δραστικά: Το πρόγραμμα αφιερώνει περισσότερο επεξεργαστικό χρόνο στο μηχανισμό του PWM και λιγότερο στις εργασίες για τις οποίες το γράψαμε. Κάπως έτσι αρχίσαμε να μιλάμε, στο τεύχος 008, για το Binary Code Modulation. Η συγκεκριμένη τεχνική υλοποιείται εύκολα και πολύ αποδοτικά, ενώ σε αρκετές περιπτώσεις μπορεί να υποκαταστήσει πλήρως το Pulse Width Modulation. Σ' αυτές περιλαμβάνεται κι εκείνη που ενθουσιάζει τους περισσότερους geeks των ηλεκτρονικών: Η ρύθμιση της φωτεινότητας μιας συστοιχίας LED.

Όλα αυτά έχουν ιδιαίτερο θεωρητικό ενδιαφέρον κι ακονίζουν τις προγραμματιστικές μας ικανότητες. Ωστόσο, η γνωριμία μας μαζί τους δεν έγινε για ακαδημαϊκούς λόγους. Πριν εξετάσουμε τα PWM και BCM είχαμε δει το κύκλωμα για τη διασύνδεση του Arduino μ' ένα 8x8 RGB LED matrix. Όπως υποψιάζεστε, λοιπόν, ο απώτερος στόχος μας ήταν ν' αποκτήσουμε τον καλύτερο δυνατό έλεγχο αυτής της διάταξης. Ένα τέτοιο matrix είναι ικανό να προβάλει εκατομμύρια χρώματα, ενώ στα 64 pixel του μπορεί κανείς να εμφανίσει μηνύματα, να σχεδιάσει απλά γραφικά (retro sprites) και να φτιάξει οτιδήποτε φανταστεί (Σ.τ.Ε. Ακόμα κι έναν δράκο, να πίνει το ποτό του ξαπλωμένος σε μια αιώρα με φόντο το σεληνιακό τοπίο;) Έτσι, για το παρόν άρθρο συνδυάσαμε το κύκλωμα από το τέταρτο μέρος της σειράς, με τις γνώσεις που αποκομίσαμε από τη μελέτη του Binary Code Modulation. Το αποτέλεσμα ήταν ιδιαίτερα ικανοποιητικό και μπορείτε να πάρετε μια γεύση από το σχετικό βίντεο στο

`http://deltahacker.gr/?p=6394`

Όμως μη νομίζετε ότι η εφαρμογή του BCM σε ένα RGB matrix επιτυγχάνεται εύκολα. Η προσπάθειά μας ήταν γεμάτη δυσκολίες. Βέβαια, ένας άνθρωπος που αγαπά τον προγραμματισμό χαίρεται όταν συναντά τέτοιου είδους δυσκολίες και πολύ περισσότερο όταν καταφέρνει και τις ξεπερνά ;) Στη συνέχεια θα παρουσιάσουμε τη λογική με την οποία αναπτύξαμε τα δικά μας προγράμματα, θα αναφέρουμε τα προβλήματα που συναντήσαμε και φυσικά τις λύσεις που δώσαμε. Ελπίζουμε η δική μας περιπέτεια να σας ωθήσει να ξεκινήσετε τη δική σας. Πολύ περισσότερο, θέλουμε να δούμε τις δικές σας υλοποιήσεις και τα δικά σας παιχνίδια!

Πρώτη προσέγγιση

Πριν αρχίσουμε να γράφουμε, αφιερώσαμε λίγο χρόνο στη μελέτη του ζητήματος και στη σχεδίαση του κώδικα. Περιττό να πούμε ότι στο τέλος το πρόγραμμά μας ήταν πολύ διαφορετικό από αυτό που είχαμε κατά νου αρχικά. Ωστόσο, το στάδιο της προετοιμασίας υπήρξε απαραίτητο για να ξεκαθαρίσουμε τα βασικά και να μπορέσουμε να ξεκινήσουμε.

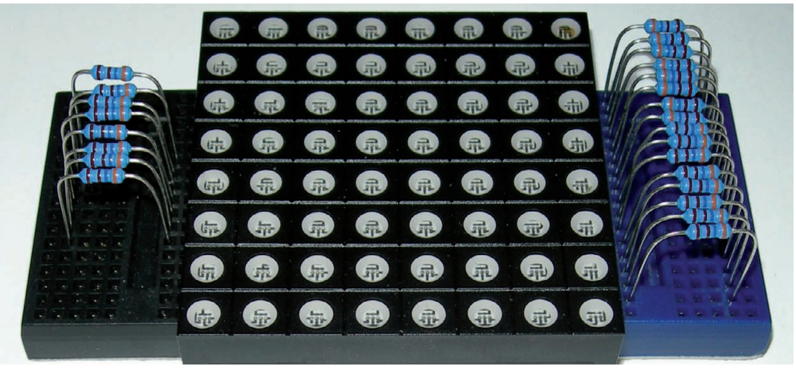
Στόχος μας ήταν να εφαρμόσουμε την τεχνική Binary Code Modulation στα LED ενός matrix. Όπως είδαμε στο προηγούμενο μέρος της σειράς (τεύχος 008), η εφαρμογή του BCM σε μερικά LED επιτυγχάνεται εύκολα. Σε γενικές γραμμές, αν το σύστημα που υλοποιούμε έχει ακρίβεια k bit[1], διαιρούμε το χρόνο της περιόδου του BCM σε k επιμέρους διαστήματα.

1. Αυτό συνεπάγεται ότι προσφέρονται 2^k επίπεδα ρύθμισης.

Στο πρώτο διάστημα εξετάζουμε την τιμή του πρώτου bit της έντασης κάθε LED και, σύμφωνα μ' αυτήν, ανάβουμε ή σβήνουμε το εκάστοτε LED. Η ίδια διαδικασία επαναλαμβάνεται και στα επόμενα υπό-διαστήματα της περιόδου, εξετάζοντας κάθε φορά το επόμενο bit της έντασης. Αυτή η περιγραφή, αν και πολύ συνοπτική, κρύβει μια σημαντική λεπτομέρεια: Για να εφαρμοστεί το BCM σε πολλά LED, ο μικροελεγκτής πρέπει να ελέγχει όλα τα LED ανά πάσα στιγμή και μάλιστα το καθένα ανεξάρτητα από τα υπόλοιπα. Ακριβώς εδώ προέκυψε η πρώτη αναποδιά: Στην περίπτωση ενός matrix, σε οποιαδήποτε χρονική στιγμή ο μικροελεγκτής ελέγχει μόνο τα LED μιας συγκεκριμένης γραμμής (ή στήλης). Επομένως, είναι αδύνατο να εφαρμόσουμε το Binary Code Modulation σε όλα τα LED του matrix -- τουλάχιστον όχι με τον τρόπο που έχουμε δει ως τώρα.

Γραμμή προς γραμμή

Η λύση βρέθηκε εύκολα. Το γεγονός ότι ο μικροελεγκτής δεν μπορεί να ελέγξει όλα τα LED του matrix ταυτόχρονα, αποτελεί πρόβλημα που έχουμε αντιμετωπίσει (και λύσει) προ πολλού. Θυμηθείτε πώς σχεδιάζουμε κάτι σ' ένα matrix. Για να πετύχουμε κάτι τέτοιο πραγματοποιούμε τη λεγόμενη σάρωση. Με απλά λόγια, ενεργοποιούμε κάθε στιγμή μια γραμμή (ή στήλη), ανάβουμε όποια LED θέλουμε επάνω της και μετά από ένα σύντομο χρονικό διάστημα προχωράμε στην επόμενη. Όταν τελειώσουν οι γραμμές του matrix (ή οι στήλες, αντίστοιχα) επιστρέφουμε στην πρώτη και επαναλαμβάνουμε τη διαδικασία. Τελικά, πραγματοποιώντας τη σάρωση με μεγάλη συχνότητα τα διάφορα LED του matrix μοιάζουν να ανάβουν ταυτόχρονα, ανεξάρτητα από τη γραμμή και τη στήλη στην οποία βρίσκονται. Ε, λοιπόν, το κολπάκι με την ταχύτατη σάρωση μπορεί να μας βοηθήσει και στην εφαρμογή του Binary Code Modulation. Δώστε βάση: Αρχικά, σκεφτόμασταν να εφαρμόσουμε το BCM επάνω σε όλα τα LED του matrix, ταυτόχρονα. Φαινόταν λογική λύση, ήταν απλή και μας βόλευε! Ωστόσο, όπως είπαμε προηγουμένως, κάτι τέτοιο είναι αδύνατο. Αυτό που μπορούμε να κάνουμε είναι να εφαρμόσουμε το BCM επάνω σε μία μόνο γραμμή του matrix, για κάποιο χρονικό διάστημα. Στη συνέχεια, θα εφαρμόσουμε την ίδια τεχνική και για το ίδιο χρονικό διάστημα στην επόμενη γραμμή του matrix και πάει λέγοντας. Με άλλα λόγια, η λύση έγκειται στην πραγματοποίηση μιας αρκετά σύνθετης σάρωσης. Σε κάθε βήμα αυτής της σάρωσης δεν θα περιοριζόμαστε σε ένα απλό άναμμα ή σβήσιμο των LED, αλλά θα ρυθμίζουμε την ένταση του καθενός! Το πρόγραμμα, δηλαδή, πρέπει να σαρώνει το matrix γραμμή προς γραμμή και σε καθεμία να εφαρμόζει την τεχνική Binary Code Modulation. Όπως καταλαβαίνετε, η λύση βρέθηκε εύκολα αλλά δεν ήταν καθόλου απλή...



Εικόνα 1
Τα μικρά RGB LED matrix διαθέτουν 64 pixel (κουκίδες) και πίσω από καθεμία περιλαμβάνουν τρία LED: Ένα κόκκινο, ένα πράσινο κι ένα μπλε LED. Έτσι, σε αυτές τις μικρές "οθόνες" κρύβονται 192 LED!

Κάθε πότε;

Έχοντας αυτόν τον οδηγό --που πιο πολύ με λαβύρινθο έμοιαζε--, ξεκίνησε η συγγραφή του κώδικα. Μη φανταστείτε ότι αρχίσαμε να γράφουμε με φούρια κι ότι μετά από λίγες ώρες είχαμε ξεμπερδέψει. Τα πράγματα δεν ήταν τόσο απλά -- ή τουλάχιστον όχι όσο νομίζαμε.

Η σάρωση του matrix θα έπρεπε να πραγματοποιείται με απόλυτα σταθερό ρυθμό, ώστε τα διάφορα γραφικά να προβάλλονται επίσης σταθερά, χωρίς flickering. Ακριβώς γι' αυτό αποφασίσαμε ότι η σάρωση του matrix θα πραγματοποιείται μέσα στο service routine ενός timer interrupt. Αυτή η σκέψη ήταν σωστή και η συγγραφή του κώδικα ξεκίνησε με το κατάλληλο σετάρισμα ενός timer. Όμως, πόσο συχνά θα έπρεπε να πυροδοτείται το σχετικό interrupt; Σε κάθε συσκευή απεικόνισης, η ελάχιστη συχνότητα σάρωσης (ή ανανέωσης) ώστε να μην παρατηρείται καθόλου flickering, καθορίζεται από το προβαλλόμενο περιεχόμενο. Σε γενικές γραμμές, όσο πληθαίνουν οι γρήγορες εναλλαγές και οι μεγάλες μεταβολές στη φωτεινότητα, τόσο μεγαλύτερη πρέπει να είναι η συχνότητα σάρωσης. Αυτό το ζήτημα όμως παραείναι θεωρητικό και το αφήσαμε για τους τεχνικούς του κινηματογράφου. Στο δικό μας project, αποφασίσαμε ότι μια απόλυτα αποδεκτή συχνότητα σάρωσης είναι τα 100Hz. Με άλλα λόγια, αποφασίσαμε ότι αν το matrix σαρώνεται εκατό φορές το δευτερόλεπτο, η εικόνα του θα είναι σταθερή θα κουράζει ελάχιστα τα μάτια και δεν θα δυσκολεύει καθόλου την κάμερα, στην περίπτωση που θα θέλαμε να φτιάξουμε ένα μικρό βίντεο :)

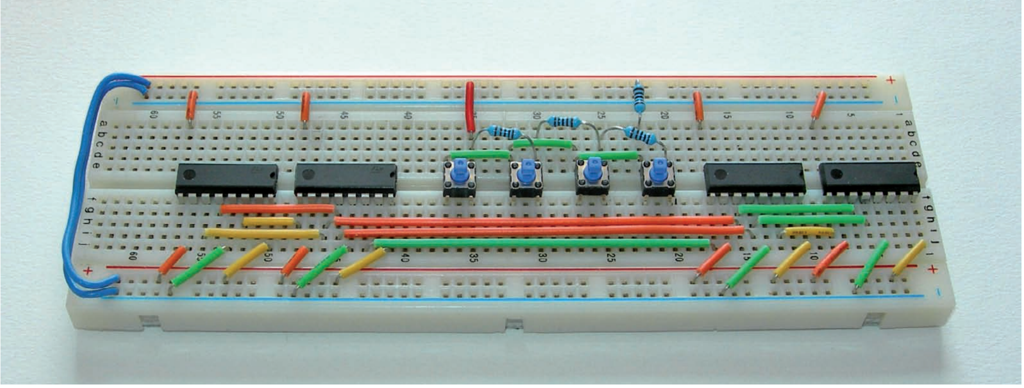
Αφού ξεκαθαρίσαμε κι αυτό το ζήτημα, αρχίσαμε επιτέλους να γράφουμε κώδικα. Το πρόγραμμα ξεκινούσε με την κατάλληλη ρύθμιση του Timer1, ώστε να προκύπτει ένα interrupt κάθε 10msec[2]. Η επιλογή του timer έγινε σχεδόν αυθαίρετα. Το ζήτημα ήταν να μην πειράξουμε καθόλου τον Timer0. Πάνω σ' αυτόν, βλέπετε, στηρίζεται η λειτουργία των διαδικασιών delay και delayMicroseconds, οι οποίες ήταν πολύ πιθανό να μας χρειαστούν αργότερα. Έχοντας επιλέξει και ρυθμίσει έναν timer, μπορούσαμε να προχωρήσουμε στο σημαντικότερο τμήμα του προγράμματος: Σε αυτό που θα πραγματοποιούσε τη σάρωση και ταυτόχρονα θα εφάρμοζε την τεχνική Binary Code Modulation σε κάθε γραμμή του matrix.

(Όχι και τόσο) λεπτομερής σχεδιασμός

Σε αυτό το σημείο, περισσότερο από ποτέ νιώσαμε την ανάγκη να βάλουμε τις σκέψεις μας σε μια σειρά. Γι' αυτό γράψαμε λίγο κώδικα σε HAPC[3], τον οποίο θα χρησιμοποιούσαμε σαν μπούσουλα κατά τη συγγραφή του αληθινού κώδικα. Ξεκινήσαμε με το γενικό πλάνο για τη ρουτίνα σάρωσης του matrix:

```
Interrupt Service Routine
   Binary Code Modulation on 1st line
   Binary Code Modulation on 2nd line
   [...]
   Binary Code Modulation on 7th line
   Binary Code Modulation on 8th line
```

2. Όταν πραγματοποιείται κάτι ανά 10msec, η συχνότητα αυτού του κάτι είναι 100Hz.
3. Highly Abstract Pseudo-Code. Πρόκειται για γλώσσα δικής μας επινόησης και προφέρεται ΧΑΨ... με κεφαλαία :P



Εικόνα 2
Το κύκλωμα με το οποίο θα παίξουμε αυτή τη φορά, θα πρέπει να σας είναι γνώριμο. Το είχαμε ξαναδεί στο 4ο μέρος της σειράς, στο deltaHacker 006. Η υλοποίηση στο breadboard ξεκινά με τους shift registers. Τα τσιπάκια τοποθετήθηκαν σε απόσταση μεταξύ τους, ώστε τα καλώδια προς το matrix να μη μπλεχτούν. Αφού συνδέσαμε την τροφοδοσία των τσιπ και μερικά ακόμα σήματα (Output Enable, master reset κ.ά.), ετοιμάσαμε και το μικρό keypad. Οι τέσσερις διακόπτες είναι συνδεδεμένοι σε έναν διαιρέτη τάσης, ώστε να ελέγχονται όλοι από μια αναλογική είσοδο του Arduino.

Στη συνέχεια, για να έχουμε καθαρό το πώς θα εφαρμόζεται το Binary Code Modulation σε κάθε γραμμή, γράψαμε λίγο ψευδοκώδικα ακόμα. Θέλαμε να υλοποιήσουμε το BCM με ακρίβεια 8bit και γι’ αυτό στο πλάνο μας συμπεριλάβαμε οκτώ βήματα: Ένα για κάθε υποδιαστήμα της περιόδου του BCM. Κατά τα γνωστά, καθένα από αυτά τα διαστήματα θα έπρεπε να έχει διπλάσια διάρκεια από το αμέσως προηγούμενό. Έτσι, το πλάνο μας διαμορφώθηκε ως εξής:

```
Binary Code Modulation on a line
   // according to 1st bit of intensity of each LED…
   Turn each LED on or off and wait for X time
   // according to 2nd bit of intensity of each LED…
   Turn each LED on or off and wait for 2 * X time
   [ … ]
   // according to 7th bit of intensity of each LED…
   Turn each LED on or off and wait for 64 * X time
   // according to 8th bit of intensity of each LED…
   Turn each LED on or off and wait for 128 * X time
```

Γράφοντας τα παραπάνω, συνειδητοποιήσαμε ότι το πρόγραμμά μας χρειαζόταν τουλάχιστον δύο frame buffers. Τα περιεχόμενα του ενός θα διαβάζονται από τη ρουτίνα σάρωσης και θα προβάλλονται στο matrix. Στο δεύτερο frame buffer θα σχεδιάζεται το επόμενο frame. Για το σκοπό αυτό ορίσαμε δύο πίνακες, των 192 bytes έκαστος. Αναρωτιέστε πώς επιλέξαμε το μέγεθός τους; Εφόσον ασχολούμαστε με ένα RGB LED matrix, κάθε pixel διαθέτει τρία επιμέρους LED (ένα κόκκινο, ένα πράσινο κι ένα μπλε). Επίσης, το BCM που θέλαμε να υλοποιήσουμε είχε ακρίβεια 8bit. Επομένως, για την περιγραφή της έντασης όλων των LED, όλων των pixel, χρειαζόμασταν 3*64 = 192 bytes.

Μια ανησυχητική διαπίστωση

Είχαμε πάρει φόρα! Διευθετούσαμε το ένα ζήτημα μετά το άλλο και νιώθαμε ότι κανένα πρόβλημα δεν μπορεί να μας σταματήσει. Μέχρι που αποφασίσαμε να υπολογίσουμε αυτό το ρημάδι το X. Μιλάμε για το χρονικό διάστημα X, το οποίο αναφέρεται σαν “X time” στο δεύτερο πλάνο του κώδικα. Αυτό το διάστημα δεν θα μπορούσε να επιλεγεί τυχαία. Σκε-

φτείτε το εξής: Πόσος χρόνος περνάει μέχρι να ολοκληρωθεί η εφαρμογή του BCM σε μία γραμμή; Αν συμβολίσουμε αυτό το μέγεθος με T, ισχύει το εξής:

```
T = X + 2*X + ... + 64*X + 128*X = 255*X
```

Προφανώς, για να ολοκληρωθεί μια σάρωση (για να εφαρμοστεί το BCM στις οκτώ γραμμές του matrix), απαιτείται χρόνος ίσος με 8T. Αυτό μας κάνει 2040X. Σημειώστε το αυτό. Όπως αναφέραμε νωρίτερα, ο χρόνος μεταξύ δύο διαδοχικών σαρώσεων είναι 10msec. Επομένως, το χρονικό διάστημα των 2040X, πρέπει να είναι (το πολύ) ίσο με 10msec.

```
2040*X = 10msec <=> X = 10msec/2040 <=> X ~= 4,9μsec
```

Ώπα! Σύμφωνα με τους υπολογισμούς μας, το χρονικό διάστημα X έπρεπε να είναι λίγο μικρότερο από 5 εκατομμυριοστά του δευτερολέπτου. Μπορούμε να γράψουμε κώδικα που θα κάνει όσα προβλέπονται από το πλάνο, τόσο γρήγορα;

Σκούρα τα πράγματα

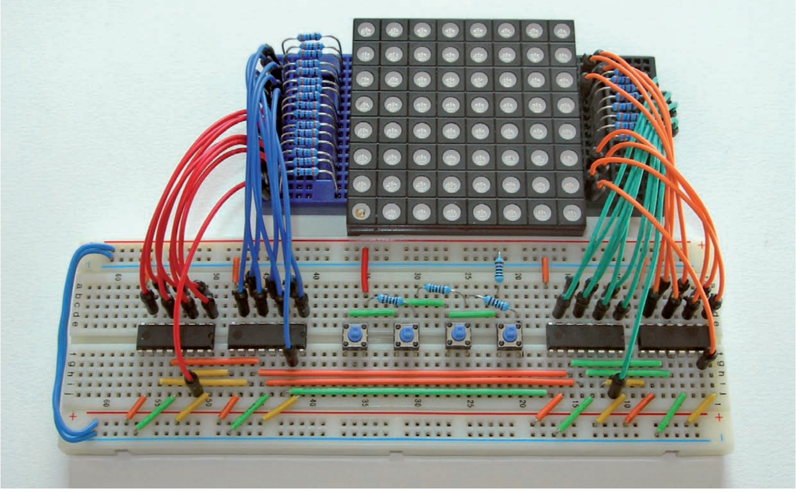
Για να απαντήσουμε σ' αυτό το ερώτημα, έπρεπε να εξετάσουμε όσα περιγράφονταν συνοπτικά (και άρα κρύβονταν) στο τελευταίο πλάνο. Έτσι, ξαναγράψαμε τον ψευδοκώδικα, αλλά με περισσότερες λεπτομέρειες. Αυτή τη φορά θέλαμε να δούμε όλες τις επιμέρους εργασίες που περιελάμβανε η εφαρμογή του BCM, σε μία γραμμή:

```
Binary Code Modulation on a line
   // according to 1st bit of intensity of each LED…
   Get intensity value for each LED of active line
   Decide the condition of each LED, according to 1st bit
   Update condition of each LED
   // according to 2nd bit of intensity of each LED…
   Get intensity value for each LED of active line
   Decide the condition of each LED, according to 2nd bit
   Update condition of each LED
   // according to 3rd bit of intensity of each LED…
   Get intensity value for each LED of active line
   Decide the condition of each LED, according to 3rd bit
   Update condition of each LED
   [ ... ]
```

Περιττό να πούμε ότι ούτε κι αυτό το πλάνο ήταν πλήρες. (Σ.τ.Ε. Αυτή η HAPC, όμως, πολύ περιγραφική -- και σίγουρα πλήρης!) Η διάρκεια κάθε ενημέρωσης των LED έπρεπε να διπλασιάζεται σε κάθε βήμα κι εμείς είχαμε παραβλέψει πλήρως αυτό το ζήτημα. Ωστόσο, μπορούσαμε ήδη να βγάλουμε ορισμένα συμπεράσματα και δεν χρειαζόταν περισσότερη ανάλυση. Ο σχεδιασμός μας ήταν αδύνατο υλοποιηθεί...

Παρατηρείστε τις εργασίες που μεσολαβούν ανάμεσα σε δύο ενημερώσεις των LED. Αρχικά, το πρόγραμμα πρέπει να ανασύρει από τη μνήμη τη φωτεινότητα για τα 24 LED της ενεργής γραμμής[4]. Επομένως, το Arduino

4. Κάθε γραμμή περιλαμβάνει 8 pixel και κάθε pixel 3 LED.



Εικόνα 3
Αμέσως μετά μπορούμε να συνδέσουμε το matrix. Η χρήση καλωδίων με σωστά επιλεγμένα χρώματα μπορεί να βοηθήσει αργότερα, κατά την αντιμετώπιση διαφόρων προβλημάτων. Επίσης, όπως φαίνεται και στη φωτογραφία, η απόσταση ανάμεσα στους shift registers αποδείχτηκε πολύ εξυπηρετική.

πρέπει να ανατρέξει στη μνήμη 24 φορές. Ο χρόνος που απαιτείται για να διαβαστούν 24 τιμές από τη μνήμη είναι 3μsec. Επίσης, πρέπει να λάβουμε υπόψη και την προετοιμασία του δείκτη, ο οποίος προσδιορίζει κάθε φορά το τι θα διαβαστεί. Αυτή η προετοιμασία δεν μπορεί να διαρκέσει λιγότερο από 1.5μsec[5]. Μετά την ανάγνωση αυτών των τιμών, το πρόγραμμα ελέγχει το δεύτερο bit καθεμίας και αποφασίζει την κατάσταση του αντίστοιχου LED. Χωρίς να μπούμε πάλι σε λεπτομέρειες, αυτή η εργασία διαρκεί κατ’ ελάχιστον 7,5μsec. Με βάση το συγκεκριμένο πλάνο, λοιπόν, ανάμεσα σε δύο ενημερώσεις των LED μεσολαβούν τουλάχιστον 12μsec. Όμως, ανάμεσα στην πρώτη και στη δεύτερη ενημέρωση των LED πρέπει να μεσολαβεί ακριβώς το χρονικό διάστημα X. Όπως θα θυμόσαστε, αυτό το διάστημα το έχουμε υπολογίσει σε λιγότερο από 5μsec. Ακόμα κι ανάμεσα στη δεύτερη και στην τρίτη ενημέρωση των LED, ο διαθέσιμος χρόνος (2*X) είναι και πάλι ανεπαρκής.

Τι σημαίνει η φράση “ενημέρωση των LED”; Το Arduino ελέγχει τα LED μέσω τεσσάρων shift registers. Επομένως, για να ενεργοποιήσει μια γραμμή και να ανάψει ή να σβήσει μερικά LED, πρέπει να στείλει 4 bytes, σειραϊκά. Ξέρετε πόσο διαρκεί αυτή η αποστολή; Στην περίπτωση που έχουμε επιλέξει τη μέγιστη ταχύτητα λειτουργίας για το SPI[6], απαιτούνται 4μsec. Έτσι, στα 12μsec που υπολογίσαμε προηγουμένως πρέπει να προσθέσουμε άλλα 4. Όπως καταλαβαίνετε, τα προβλήματα στο σχεδιασμό του προγράμματός μας έχουν πάψει να είναι σοβαρά. Έχουν γίνει σχεδόν κωμικά!

Όλη αυτή την ώρα έχουμε πέσει με τα μούτρα στη σάρωση του matrix. Λογικό, θα σκεφτεί κανείς, αφού αυτή η εργασία είναι μακράν πιο πολύπλοκη από κάθε άλλη. Είναι εξίσου λογικό, όμως, να προβληματιστούμε λίγο με την προετοιμασία του επόμενου frame. Αν εξαντλήσουμε τα 10msec που μεσολαβούν ανάμεσα σε δύο σαρώσεις, το Arduino δεν θα προλάβει ποτέ να σχεδιάσει το επόμενο frame. Το εξαιρετικά πολύπλοκο πρόγραμμά μας (αν κι όταν καταφέρουμε να το ολοκληρώσουμε) θα καταντήσει

5. Αυτό ισχύει στην ιδανική περίπτωση, όταν δηλαδή οι προς ανάγνωση θέσεις βρίσκονται σε διαδοχικές διευθύνσεις. Τότε, ανάμεσα στις διαδοχικές αναγνώσεις ο δείκτης αυξάνεται κατά μία μονάδα κι αυτή η μεταβολή είναι η απλούστερη δυνατή.

6. Το κύκλωμα που στέλνει σειραϊκά τα δεδομένα στους shift register.

να δείχνει μια στατική εικόνα. Για να αποφύγουμε αυτή την τραγική κατάληξη, πρέπει να δεσμεύσουμε ένα μέρος των 10msec για τη σχεδίαση του επόμενου frame. Όπως αντιλαμβάνεστε, ο χρόνος που απαιτείται γι' αυτή την εργασία ποικίλει. Ένα πρόγραμμα που προβάλει μερικά sprites, μπορεί να σχεδιάζει κάθε frame σε ελάχιστο χρόνο. Αντίθετα, ένα παιχνίδι χρειάζεται περισσότερο χρόνο. Κατά μέσο όρο, θα λέγαμε ότι ο εν λόγω χρόνος ανέρχεται στα 4msec[7]. Ωστόσο, αν θεωρήσουμε αυτό το χρόνο ως δεσμευμένο και δώσουμε στη σάρωση μόνο τα εναπομείναντα 6msec, ο χρόνος X θα μειωθεί περαιτέρω. Για την ακρίβεια, θα πέσει στα 2,9μsec και τα ήδη ασφυκτικά περιθώρια θα στενέψουν κι άλλο!

Ασυμπτωτική προσέγγιση

Ήταν φανερό ότι το Arduino δεν προλάβαινε να ολοκληρώσει τις απαιτούμενες εργασίες, ούτε κατά διάνοια. Μπροστά σε αυτό το πρόβλημα, η πρώτη λύση που σκεφτήκαμε ήταν να ελαττώσουμε τη συχνότητα σάρωσης. Με αυτόν τον τρόπο θα εξασφαλίζαμε περισσότερο χρόνο για κάθε σάρωση και κατ' επέκταση ο χρόνος X θα αυξανόταν. Θα άξιζε όμως κάτι τέτοιο; Τα 100Hz που είχαμε επιλέξει σαν συχνότητα σάρωσης ήταν σχεδόν οριακά. Τα LED αποτελούν ταχύτατες και πολύ έντονες πηγές φωτός, με αποτέλεσμα το flickering να γίνεται εύκολα αντιληπτό. Εξάλλου, ακόμα κι αν μειώναμε τη συχνότητα σάρωσης στα 50Hz, δεν θα καταφέρναμε να λύσουμε το πρόβλημα. Σε αυτήν την περίπτωση ο χρόνος μεταξύ δύο σαρώσεων θα διπλασιαζόταν (20msec) και το ίδιο θα συνέβαινε στο χρόνο X (9,8μsec). Παρ' όλα αυτά, ο χρόνος X θα παρέμενε μικρότερος από τα 16μsec, που απαιτούσαν οι εργασίες ανάμεσα σε δύο ενημερώσεις των LED.

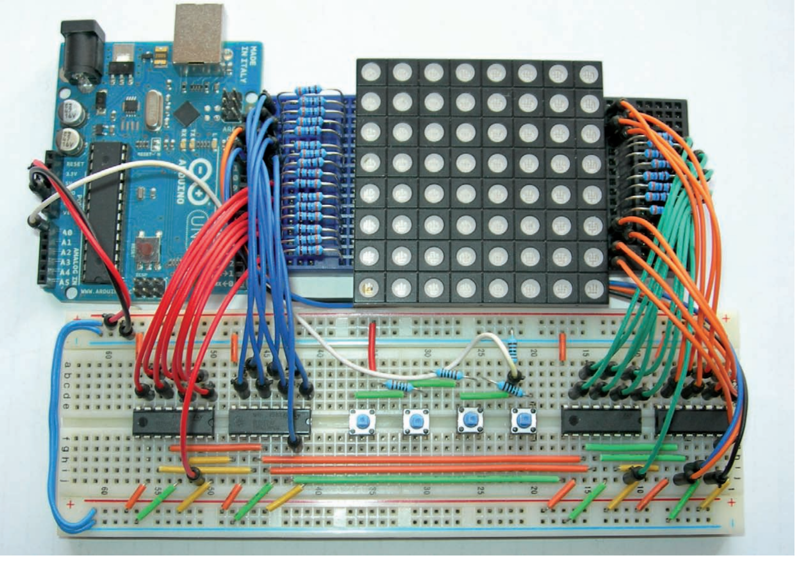
Φαινόταν αδύνατο να αυξήσουμε επαρκώς το χρόνο X, αλλά δεν ήμασταν διατεθειμένοι να παραιτηθούμε. Ξέρετε τι λένε; Όταν δεν πάει ο Μωάμεθ στο βουνό, πάει το βουνό στο Μωάμεθ. Έτσι, η επόμενη λύση που σκεφτήκαμε ήταν να περιορίσουμε το πλήθος των εργασιών που μεσολαβούσαν ανάμεσα σε δύο ενημερώσεις των LED. Με αυτόν τον τρόπο δεν θα χρειαζόταν να αυξήσουμε καθόλου το χρόνο X. Σκεφτείτε το εξής: Αν προετοιμάσουμε τα byte ελέγχου που πρέπει να στείλουμε κάθε φορά στους shift registers, οι ενημερώσεις των LED θα μπορούν να πραγματοποιούνται ταχύτερα. Δείτε τις ακόλουθες γραμμές ψευδοκώδικα και θα καταλάβετε:

```
Binary Code Modulation on a line
    // according to 1st bit of intensity of each LED…
    Get precalculated LED states
    Update condition of each LED
    // according to 2nd bit of intensity of each LED…
    Get precalculated LED states
    Update condition of each LED
    // according to 3rd bit of intensity of each LED…
    Get precalculated LED states
    Update condition of each LED
    [ ... ]
```

7. Μην προβληματίζεστε. Αυτή η εκτίμηση αποτελεί προϊόν εμπειρίας κι εξάλλου δεν είναι καθόλου κρίσιμη.

Κατά τα γνωστά, κάθε ενημέρωση των LED πρέπει να ισχύει για το διπλάσιο χρόνο από την προηγούμενη. Κάτι τέτοιο θα μπορούσε να γίνει εύκολα με την προσθήκη των κατάλληλων καθυστερήσεων. Ωστόσο, για λόγους απλότητας κρατήσαμε τον ψευδοκώδικα στην απλούστερη δυνατή μορφή. Όπως βλέπετε, ανάμεσα σε κάθε ενημέρωση των LED μεσολαβούσαν πλέον λιγότερες εργασίες. Συγκεκριμένα, μεσολαβούσε μόνο η ανάγνωση τεσσάρων bytes από τη μνήμη και η αποστολή τους στους shift registers. Αν φροντίζαμε ώστε αυτά τα bytes να βρίσκονται σε διαδοχικές θέσεις της μνήμης, τότε η λήψη και η αποστολή τους θα διαρκούσε ακριβώς 4,75μsec. Αυτός ο χρόνος, έστω και οριακά, είναι μικρότερος από το χρόνο X (4,9μsec). Είχαμε ιδρώσει, αλλά τα είχαμε καταφέρει! Ή μήπως όχι; Η συγκεκριμένη αλλαγή στην εφαρμογή του BCM προϋπέθετε ότι οι καταστάσεις των LED (για κάθε bit της έντασης του καθενός) έχουν ήδη υπολογιστεί. Αυτό σήμαινε ότι έπρεπε να τροποποιήσουμε και τη γενικότερη μορφή της ρουτίνας σάρωσης. Για να ξεκαθαρίσουμε τη μορφή της ανανεωμένης ρουτίνας σάρωσης, γράψαμε και πάλι λίγες γραμμές ψευδοκώδικα:

```
Interrupt Service Routine
   for k = 1 to 8
      calculate LED states of line k, according to bit 1 of
intensity
      [ ... ]
      calculate LED states of line k, according to bit 6 of
intensity
   Binary Code Modulation on 1st line
   Binary Code Modulation on 2nd line
   [ ... ]
   Binary Code Modulation on 6th line
   Binary Code Modulation on 7th line
```



Εικόνα 4
Το ολοκληρωμένο κύκλωμα περιλαμβάνει και το Arduino. Η σύνδεσή του με το υπόλοιπο κύκλωμα έχει ελάχιστες συνδέσεις. Οι shift registers τροφοδοτούνται με δεδομένα από το κύκλωμα SPI και για τη σχετική διασύνδεση απαιτούνται μόνο τρία καλώδια. Ο διαιρέτης τάσεων με τους διακόπτες συνδέεται με ένα καλώδιο στην αναλογική είσοδο A0. Τέλος, χρειάζονται δύο ακόμα καλώδια, για την τροφοδοσία του breadbord από το Arduino.

Αυτό ήταν όλο. Η ρουτίνα σάρωσης υπολογίζει την κατάσταση όλων των LED, για καθένα από τα bit της έντασης. Προφανώς, αν και αυτό δεν φαίνεται στο πλάνο, οι σχετικές τιμές θα τοποθετούνταν σε διαδοχικές θέσεις της μνήμης, ώστε η ανάκτησή τους κατά την εφαρμογή του BCM να είναι ταχύτατη. Μήπως τώρα τα είχαμε καταφέρει; Δυστυχώς όχι! Βλέπετε, εξαιτίας αυτής της δομικής αλλαγής στη μορφή της ρουτίνας σάρωσης, είχε αλλάξει η τιμή του X. Χωρίς να μπούμε σε κουραστικές λεπτομέρειες, θα σας πούμε ότι οι εργασίες που είχαμε προσθέσει στην αρχή της ρουτίνας σάρωσης διαρκούσαν 1320μsec. Επομένως, ο διαθέσιμος χρόνος για την εφαρμογή του BCM στις γραμμές του matrix μειωνόταν. Συγκεκριμένα, από τα 10msec έπεφτε στα 8,68msec. Ως εκ τούτου, μειωνόταν και η τιμή του χρόνου X σε 4,25μsec. Νταρν! (Σ.τ.Ε. Η λέξη στο πρωτότυπο κείμενο ήταν "διάβολε", αλλά τη λογοκρίναμε.) Ο χρόνος X γινόταν (ξανά) μικρότερος από το χρόνο που απαιτούσαν οι εργασίες μεταξύ των ενημερώσεων των LED (4,75μsec). Βέβαια, οι αλλαγές που είχαμε κάνει δεν ήταν εντελώς άσκοπες. Σκεφτείτε ότι πριν από αυτές, ο χρόνος X ήταν στα 4,9μsec, ενώ οι εργασίες απαιτούσαν 16μsec. Με την τωρινή μορφή της ρουτίνας σάρωσης, η απόκλιση του X από την επιθυμητή ήταν μικρότερη από 1μsec!

Τακτική υποχώρηση

Το πρόβλημα με το χρόνο X είχε περιοριστεί, αλλά ξεχνούσαμε κάτι βασικό. Πότε θα σχεδιάζονταν τα νέα frame; Οι υπολογισμοί μας βασίζονταν στην υπόθεση ότι η ρουτίνα σάρωσης εξαντλεί όλο το χρόνο μεταξύ δύο σαρώσεων. Ωστόσο, όπως είπαμε νωρίτερα, θα έπρεπε να ξεκλέψουμε τουλάχιστον 4msec για τη σχεδίαση των frame. Αν κάναμε κάτι τέτοιο, ο χρόνος για τη σάρωση του matrix θα μειωνόταν από τα 10msec στα 6msec. Βέβαια, ακόμα και με τη βελτιωμένη μορφή της ρουτίνας σάρωσης, κάτι τέτοιο ήταν αδύνατο. Ο χρόνος X ήταν ήδη μικρότερος από τον αναγκαίο. Τότε, φάνηκε ότι έπρεπε να προχωρήσουμε σε πιο δραστικά μέτρα. Ο χρόνος σάρωσης του matrix έπρεπε να μειωθεί οπωσδήποτε! Όμως ποιος παράγοντας επηρέαζε περισσότερο τη διάρκεια αυτής της διαδικασίας; Όπως είπαμε παραπάνω, οι πράξεις στην αρχή της ρουτίνας σάρωσης διαρκούσαν 1,32msec. Επομένως, τα υπόλοιπα 8,68msec καταλαμβάνονταν από την εφαρμογή του BCM σε κάθε γραμμή του matrix. Με άλλα λόγια, η εφαρμογή του BCM απαιτούσε το 86% του διαθέσιμου χρόνου...

Η λύση βρισκόταν μπροστά μας. Η ιδέα της έκπτωσης από τους αρχικούς στόχους δεν μας άρεσε, αλλά ήμασταν υποχρεωμένοι να την ακολουθήσουμε: Έπρεπε να περιορίσουμε την ακρίβεια του Binary Code Modulation. Μειώνοντας την ακρίβεια του BCM στα 6bit, η εφαρμογή του σε κάθε γραμμή θα ολοκληρωνόταν σε έξι βήματα κι όχι σε οκτώ. Αυτή η μικρή αλλαγή θα ασκούσε τεράστια επίδραση στη διάρκεια της εφαρμογής του BCM. Βλέπετε, κάθε ενημέρωση των LED ισχύει για το διπλάσιο χρονικό διάστημα, σε σχέση με την αμέσως προηγούμενη. Με άλλα λόγια, το πλήθος των bit της ακρίβειας του BCM κι ο συνολικός χρόνος για την εφαρμογή του σε μια γραμμή, συνδέονται εκθετικά! Συγκεκριμένα, αν αυτός ο χρόνος συμβολίζεται με T και το πλήθος των bit με το k, ισχύει η ακόλουθη σχέση:

```
T = X * (2^k - 1)
```

Επομένως, αν μειώσουμε το πλήθος των bit κατά 2, ο χρόνος εφαρμογής του BCM σε μια γραμμή υποτετραπλασιάζεται. Μετά από αυτή τη μικρή

υποχώρηση στην ακρίβεια του BCM, τα προβλήματά μας θα λυθούν οριστικά!

Μανία βελτιστοποίησης

Μετά από όλη αυτήν την ταλαιπωρία μπορούσαμε επιτέλους να γράψουμε λίγο πραγματικό κώδικα. Με τη βοήθεια των πλάνων που είχαμε καταστρώσει, η συγγραφή της ρουτίνας σάρωσης έγινε γρήγορα και χωρίς προβλήματα. Κάπως έτσι περάσαμε στο επόμενο στάδιο: Το πρόγραμμά μας μπορούσε πλέον να «οδηγήσει» το matrix κι έπρεπε να φροντίσουμε για τον σχεδιασμό των frames. Γι' αυτή την εργασία είχαμε προβλέψει περί τα 4msec, ανάμεσα στις σαρώσεις του matrix. Ωστόσο, ο σχεδιασμός ενός πολύπλοκου frame θα μπορούσε να διαρκέσει περισσότερο. Ακριβώς γι' αυτό, ο κύριος βρόχος του προγράμματος πήρε την ακόλουθη μορφή:

```
main program loop
    draw_new_frame()
    update_buffers()
    react_to_keys()
```

Ο κύριος βρόχος του προγράμματος καλεί τρεις συναρτήσεις. Νομίζουμε ότι η αποστολή καθεμιάς είναι προφανής. Η σχεδίαση ενός frame μπορεί να διαρκεί οσοδήποτε. Ακόμα και μία ώρα! Η αντιγραφή του "πρόχειρου" frame buffer στο "κανονικό" πραγματοποιείται *μόνον* όταν ολοκληρωθεί η σχεδίαση του frame. Μετά από αυτή την αντιγραφή, η ρουτίνα σάρωσης θα προβάλει πλέον το νέο frame. Τα πράγματα εδώ είναι τόσο απλά, που καταντούν βαρετά. Εμείς όμως, αφού είχαμε παιδευτεί υπερβολικά για να κόψουμε λίγα μικροδευτερόλεπτα από 'δω και λίγα από 'κει, αναζητούσαμε ασυναίσθητα την επόμενη βελτιστοποίηση. Εξετάζοντας τον κώδικα, δεν αργήσαμε να βρούμε τον επόμενο στόχο. Η αντιγραφή των frame buffer περιελάμβανε έναν βρόχο, ο οποίος διάβαζε 192 bytes από τη μνήμη[8] και τα αποθήκευε πάλι εκεί, αλλά σε διαφορετική θέση. Μετά από μετρήσεις που πραγματοποιήσαμε, διαπιστώσαμε ότι η συγκεκριμένη εργασία διαρκούσε περί τα 216μsec. Όπως αντιλαμβάνεστε, αυτό το χρονικό διάστημα δεν εισήγαγε καμία αισθητή καθυστέρηση στη λειτουργία του προγράμματος. Εμείς, ωστόσο, νιώθαμε την ανάγκη να την απαλείψουμε!

Η αντιγραφή των στοιχείων ενός πίνακα σε έναν άλλον αποτελεί μια απλούστατη διαδικασία. Πόσο αποδοτικότερα θα μπορούσε να γραφτεί; Όταν συναντάμε προβλήματα που σχετίζονται με τη μαζική μεταφορά δεδομένων, το προγραμματιστικό μας ένστικτο πηγαίνει αυτόματα στους δείκτες. Αυτά τα --ομολογουμένως δύσχρηστα-- μεγέθη, επιτρέπουν μερικά πολύ έξυπνα και αποδοτικά κολπάκια. Ε, λοιπόν, ένα από αυτά επιστρατεύσαμε κι εμείς για να επιταχύνουμε την αντιγραφή των frame buffers. Συγκεκριμένα, ξεκινήσαμε ορίζοντας τους εξής δύο δείκτες:

```
char * fb_ptr_cur = frame_buffer1;
char * fb_ptr_new = frame_buffer2;
```

Στη συνέχεια, φροντίσαμε ώστε όλες οι αναφορές που γίνονταν μέσα στη ρουτίνα σάρωσης προς το frame buffer, να πραγματοποιούνται μέσω του δείκτη fb_ptr_cur. Αντίστοιχα, φροντίσαμε ώστε όλες οι αναφορές που γίνονταν μέσα στη ρουτίνα σχεδίασης του νέου καρέ προς το αντίστοιχο

8. Επειδή κάθε frame buffer αντιστοιχεί σ' έναν πίνακα των 192byte...

frame buffer, να πραγματοποιούνται μέσω του δείκτη fb_ptr_new. Μετά κι απ' αυτό, απομακρύναμε το βρόχο των 192 επαναλήψεων ο οποίος αντέγραφε τα δεδομένα των frame buffers, και στη θέση του βάλαμε τις ακόλουθες γραμμές:

```
fb_ptr_tmp = fb_ptr_cur;
fb_ptr_cur = fb_ptr_new;
fb_ptr_new = fb_ptr_tmp;
```

Ο δείκτης fb_ptr_tmp ήταν προσωρινός και χρησίμευε μόνο για την αντιμετάθεση των τιμών, των δύο άλλων δεικτών. Όπως αντιλαμβάνεστε, με αυτό το κολπάκι η ρουτίνα σάρωσης θα διάβαζε πλέον το frame buffer με το νέο καρέ, ενώ η ρουτίνα σχεδίασης του νέου καρέ θα άρχιζε να γράφει πάνω στο frame buffer με το παλιό καρέ. Ουσιαστικά, με την αντιμετάθεση των δεικτών καταφέρναμε να αντιστρέψουμε τους ρόλους των δύο frame buffers. Έτσι, η δουλειά μας γινόταν εξίσου καλά και μάλιστα σε λιγότερο από ένα μικροδευτερόλεπτο! Η βελτίωση που είχαμε πετύχει ξεπερνούσε το 216% :)

Ανάπτυξη δίχως τέλος

Αφού εφαρμόσαμε και το κολπάκι που περιγράψαμε παραπάνω, η μανία βελτιστοποίησης άρχισε να μας εγκαταλείπει. Ή τουλάχιστον έτσι νομίσαμε! Το πρόγραμμα που είχαμε μπροστά μας ήταν σχεδόν ολοκληρωμένο. Είχαμε κατασκευάσει μια μηχανή η οποία σάρωνε το matrix και φρόντιζε για τον κατάλληλο (και αποδοτικό!) χειρισμό των frame buffers. Αυτό που έμενε ήταν γράψουμε διάφορες εκδοχές για τη ρουτίνα σχεδίασης των νέων καρέ. Για παράδειγμα, για να φτιάξουμε το αγαπημένο μας ραλάκι, αρκούσε μια ρουτίνα η οποία θα σχεδίαζε το χάρτη, θα τοποθετούσε το όχημά μας και θα το μετακινούσε σύμφωνα με τα πατήματα των πλήκτρων. Η ίδια ρουτίνα θα έλεγχε αν το όχημά μας έχει πέσει πάνω σε κάποιο εμπόδιο και πάει λέγοντας. Νομίζουμε ότι το όλο σκεπτικό είναι ξεκάθαρο. Η μηχανή που είχαμε κατασκευάσει αποτελούσε τη βασική υποδομή για να φτιάξουμε οτιδήποτε γεννούσε η φαντασία μας! Σε αυτό το στάδιο, ένας φυσιολογικός άνθρωπος θα ξεκινούσε να αναπτύσσει με χαρά ένα demo effect, ένα παιχνιδάκι ή ένα πρόγραμμα εμφάνισης κυλιόμενων μηνυμάτων, πάνω στο έγχρωμο matrix. Εμείς κάναμε κάτι διαφορετικό...

Η υπόσχεση

Πριν ξεκινήσουμε την περιγραφή του όλου εγχειρήματός, αναφέραμε ότι ένα RGB LED matrix είναι ικανό να προβάλει εκατομμύρια χρώματα. Σας δώσαμε μια υπόσχεση, έτσι δεν είναι; Εμείς πάντως έτσι νιώθαμε! Επομένως, όταν συνειδητοποιήσαμε ότι την είχαμε αθετήσει, σπεύσαμε να επανορθώσουμε. Έχοντας μειώσει την ακρίβεια του Binary Code Modulation στα 6bit, το χρώμα κάθε pixel περιγραφόταν πλέον από 18bit. Αυτό σήμαινε ότι το πρόγραμμα μπορούσε να προβάλει 2^18 ή αλλιώς 262.144 χρώματα. Για να είμαστε εντάξει απέναντί σας, έπρεπε να αυξήσουμε την ακρίβεια του BCM τουλάχιστον στα 7bit. Με αυτόν τον τρόπο το χρώμα καθενός pixel θα περιγραφόταν από 21bit και το πρόγραμμά μας θα μπορούσε να προβάλει 2^21 ή 2.097.152 χρώματα! Όπως υποψιάζεστε, μετά από αυτές τις σκέψεις επιστρέψαμε πάλι στη ρουτίνα σάρωσης. Έπρεπε να επιταχύνουμε όλες τις εργασίες και να ξεκλέψουμε λίγο ακόμα χρόνο. Αυτή τη φορά τα πράγματα είχαν ζορίσει για τα καλά. Η ρουτίνα ήταν δομημένη με τον καλύτερο τρόπο που μπορούσαμε να σκεφτούμε και η

υλοποίησή της είχε γίνει σε C, με τον πιο αποδοτικό τρόπο που μπορούσαμε να πετύχουμε. Τι θα μπορούσε να γίνει άραγε;

Ένας παρανοϊκός θα διέγραφε το πρόγραμμα και θα το έγραφε πάλι από την αρχή, εξ ολοκλήρου σε Assembly! Εμείς αρκεστήκαμε σε μια ενδιάμεση λύση: Αποφασίσαμε να γράψουμε σε Assembly μόνο τη ρουτίνα σάρωσης :D Ως γνωστόν, το περιβάλλον ανάπτυξης του Arduino χρησιμοποιεί τον compiler GCC. Κάτι ακόμα που είναι γνωστό, είναι πως ο GCC επιτρέπει το λεγόμενο ASM inlining. Με απλά λόγια, επιτρέπει την ενσωμάτωση κώδικα γραμμένου σε Assembly, μέσα σε προγράμματα γραμμένα σε C. Οι λεπτομέρειες για το πώς μπορεί να γίνει κάτι τέτοιο ξεφεύγουν από το θέμα μας. Αρκεί να σας πούμε ότι θα μπορούσαν να αποτελέσουν το θέμα για ένα ολόκληρο άρθρο[9]. Για να πετύχουμε το στόχο μας χρειάστηκε άφθονη μελέτη, πολλά πειράματα κι ακόμη περισσότερα λάθη. Στο τέλος όμως τα καταφέραμε! Μεταξύ μας, πέρα από την ανάγκη που νιώθαμε να κρατήσουμε την υπόσχεσή μας, θέλαμε να πετύχουμε κάτι ακόμα: Θέλαμε να δείξουμε την ανωτερότητα που έχει ο κώδικας σε Assembly, όταν έχει γραφτεί και βελτιστοποιηθεί "χειροκίνητα". Για την ιστορία, λοιπόν, σημειώστε ότι η αρχική εκδοχή της ρουτίνας σάρωσης απαιτούσε 7,2msec και υλοποιούσε το BCM με ακρίβεια 6bit. Η τελική εκδοχή, με τη βοήθεια της Assembly, απαιτούσε 6,8msec και προσέφερε ακρίβεια 7bit. Το πρόγραμμά μας μπορούσε επιτέλους να προβάλει εκατομμύρια χρώματα κι εμείς, εκτός από χαρούμενοι, ήμασταν κι εξαντλημένοι.

Παιχνίδι με τον κώδικα

Όλες αυτές οι περιγραφές και οι αναλύσεις δεν θα μπορούσαν να μείνουν στο χαρτί. Όπως και κάθε άλλη φορά, λοιπόν, θα σας δώσουμε ένα πακέτο με όλα τα προγράμματα που κατασκευάσαμε. Μπορείτε να το πάρετε από εδώ:

http://bit.ly/dh012ArduinoSrc

Στο πακέτο θα βρείτε τα προγράμματα που τρέξαμε στο Arduino για να φτιάξουμε το βιντεάκι μας (http://deltahacker.gr/?p=6394), καθώς και δύο templates. Πρόκειται για τις δύο βασικές εκδοχές της μηχανής μας: Μια γραμμένη εξ ολοκλήρου σε C και μία στην οποία περιλαμβάνεται κώδικας σε Assembly. Εσείς μπορείτε να χρησιμοποιήσετε οποιαδήποτε από τις δύο θέλετε, για ν' αναπτύξετε τις δικές σας εφαρμογές. Για να σχεδιάσετε οτιδήποτε στο matrix, αρκεί να το σχεδιάσετε στο πρόχειρο frame buffer, χρησιμοποιώντας το δείκτη fb_ptr_new. Για παράδειγμα, για να γράψετε κάτι στη θέση index του εν λόγω frame buffer, αρκεί να δώσετε κάτι τέτοιο:

```
(fb_ptr_new + index) = ...
```

Όλα αυτά θα πρέπει να γίνουν στη συνάρτηση draw_new_frame(). Τέλος, αν η εφαρμογή σας αλληλεπιδρά με το χρήστη, ο κώδικας για την αντίδραση του προγράμματος σε κάθε πάτημα πλήκτρου πρέπει να μπει στη συνάρτηση react_on_keys(). Σημειώστε ότι η μεταβλητή keypad_value φέρει κάθε στιγμή μια τιμή από το διάστημα 0 έως 4. Όταν έχει την τιμή μηδέν δεν έχει πατηθεί κανένα πλήκτρο, ενώ οι υπόλοιπες τιμές αντιστοιχούν στο πάτημα ενός από τα τέσσερα κουμπάκια του κυκλώματός μας.

Κάπου εδώ σας αφήνουμε -- κι ελπίζουμε να δούμε σύντομα τις δικές σας εφαρμογές!

9. Ε; Τι λέτε;

ΕΠΙΘΕΣΗ ΣΤΑ ΤΥΦΛΑ!

(Μέρος 2ο)

Έχουμε στήσει το τέλειο θύμα: Ένα web application το οποίο δεν πραγματοποιεί σωστό input sanitization, με συνέπεια να προσφέρει ένα (ορθάνοιχτο) injection point. Έτσι, μπορούμε να μελετήσουμε ένα είδος επιθέσεων SQL injection, το οποίο αποσπά κρυφές πληροφορίες από τη βάση με τον πιο πονηρό και ευρηματικό τρόπο...



Skill: Intermediate

Tags: blind SQL injection, ASCII codes

by 41i3n

Στο άρθρο που αρχίζει από τη σελίδα 54 υλοποιήσαμε ένα απλοϊκό web application και τώρα είμαστε έτοιμοι να του επιτεθούμε ανελέητα. Και όταν λέμε ανελέητα, το εννοούμε! Αρκεί να αναλογιστείτε πως για την ανακάλυψη ενός κρυπτογραφημένου συνθηματικού θα χρειαστούν 512 αιτήματα HTTP. Στην πορεία, βέβαια, θα δούμε ότι αυτό διορθώνεται. Ωστόσο, οι επιθέσεις blind SQL injection περιλαμβάνουν συνήθως *πολλά* HTTP requests. Ευτυχώς, στις δοκιμές μας δεν θα συναντήσουμε εμπόδια του τύπου Intrusion Detection Systems (IDS) ή Web Application Firewalls (WAF), όπως θα συνέβαινε σε κανονικούς servers. Αυτή η διευκόλυνση θα μας επιτρέψει να εστιάσουμε στη λογική της επίθεσης και να την κατανοήσουμε καλύτερα. Είσαστε έτοιμοι; Ξεκινάμε!

Injection point

Εφόσον ήταν όλα έτοιμα, ανοίξαμε έναν browser κι αρχίσαμε τις δοκιμές. Θα έχετε προσέξει (κι αν όχι, σας το λέμε εμείς τώρα) ότι αλλάζοντας την τιμή του id, σ' ένα URL στην address bar, μεταφερόμαστε στο αντίστοιχο άρθρο χωρίς να κάνουμε κλικ σε κάποιο σύνδεσμο. Έτσι, αν στο address bar δώσουμε κάτι τέτοιο:

```
http://localhost/webapp.php?id=1
```

θα μεταφερθούμε στο άρθρο με id = 1, το οποίο παρεμπιπτόντως ανήκει στο χρήστη subZraw. Αν στο id αποδίδαμε την τιμή 2, τότε θα εμφανιζόταν το άρθρο του Thiseas και πάει λέγοντας. Τι θα γινόταν, άραγε, αν θέταμε το id ίσο με 300; Ωπ! Τότε θα προέκυπτε ένα μεγαλοπρεπέστατο "Page not found". Από αυτό και μόνο, μπορεί να αντιληφθεί κανείς ότι το input sanitization είναι (τουλάχιστον) προβληματικό. Κανονικά, η τιμή του id θα έπρεπε να ελέγχεται κι όταν βρίσκεται εκτός των αποδεκτών ορίων να απορρίπτεται. Επιστρέφοντας στο θέμα μας --τις επιθέσεις blind SQL injection--, μετά από αυτή τη διαπίστωση θα ακολουθούσε μια δοκιμή της ακόλουθης μορφής:

```
http://localhost/webapp.php?id=1 AND 1=1
```

Δοκιμάζοντας το συγκεκριμένο URL, το web application εμφανίζει το άρθρο του subZraw. Αυτό μπορεί να φαίνεται παράξενο, αλλά σκεφτείτε το εξής: Το id=1 αποτελεί ένα valid query, εφόσον υπάρχει άρθρο με αυτό το id. Εξάλλου, η ισότητα 1=1 είναι πάντα αληθής. Επομένως, ολόκληρο το query που στείλαμε (id=1 AND 1=1) αποτιμάται ως TRUE και η βάση επιστρέφει αυτό που της ζητήσαμε κανονικά. Εν ολίγοις, το query που της στείλαμε δεν ήταν FALSE και γι' αυτό το εξυπηρέτησε χωρίς παράπονα. Μήπως όμως το input sanitization έκοψε το δεύτερο σκέλος του query μας; Μπορεί να φτιάξαμε μόνοι μας το συγκεκριμένο web application και να γνωρίζουμε ότι δεν συμβαίνει κάτι τέτοιο, αλλά σε ένα πιο ρεαλιστικό σενάριο δεν θα είχαμε ιδέα! Για να διαπιστώσουμε αν το input sanitization κόβει τα queries και δεν επιτρέπει την εκτέλεση λογικών πράξεων, μπορούμε να κάνουμε την εξής δοκιμή:

```
http://localhost/webapp.php?id=1 AND 1=0
```

Η σελίδα που εμφανίζεται περιέχει το μήνυμα "Page not found". Κάπου εδώ, αν είχαμε απέναντί μας έναν πραγματικό στόχο θα μπορούσαμε ν' αρχίσουμε τους πανηγυρισμούς. Το γεγονός ότι η απάντηση της βάσης ήταν αρνητική, ενώ γνωρίζουμε ότι υπάρχει άρθρο με id=1, συνεπάγεται ότι το query μας *δεν* κόπηκε. Έτσι, η βάση εκτίμησε το 1=0 ως FALSE

(διότι είναι) κι εξαιτίας της λογικής πράξης AND αποτίμησε κι ολόκληρο το query ως FALSE. Κατά συνέπεια, δεν επέστρεψε το άρθρο με id=1, αλλά μιαν αρνητική απάντηση :)

Όπως αντιλαμβάνεστε, τα injection points δεν είναι πάντοτε τόσο απλοϊκά. Το δικό μας web application όμως αποτελεί ξέφραγο αμπέλι! Ωστόσο, οι πιθανότητες επιτυχίας μιας προσπάθειας μπορούν να αυξηθούν, όσο πιο καλά γνωρίζουμε τη γλώσσα SQL. Κάπου εδώ θα διαμαρτυρηθείτε: Μα, η γλώσσα SQL δεν είναι ίδια για όλες τις βάσεις! Έτσι είναι. Γι' αυτό και θα πρέπει πρώτα να μαθαίνουμε το είδος της βάσης δεδομένων, που χρησιμοποιεί ο εκάστοτε server. Πριν ξεκινήσουμε τις δοκιμές, κανονικά θα έπρεπε να προηγηθεί ένα information gathering, με σκοπό να ανακαλύψουμε το SQL engine αλλά και την έκδοσή του. Στη συνέχεια, θα έπρεπε να αφιερώσουμε αρκετό χρόνο, ώστε να εξοικειωθούμε με τα queries για τη συγκεκριμένη βάση...

Συμβάσεις...

Στο εξής --κι επειδή ο text editor δημιουργεί hyperlink σε κάθε γραμμή κι έχει αρχίσει να μου σπάει τα νεύρα-- θα αντικαταστήσουμε το πρώτο (και σταθερό) τμήμα των αιτημάτων μας με ένα άλλο string. Συγκεκριμένα, ολόκληρο το request "http://localhost/webapp.php?id=1" θα αναφέρεται σαν "web_app_url". Για παράδειγμα, το ακόλουθο request

```
http://localhost/webapp.php?id=1 AND 1=1
```

θα αναφέρεται ως εξής:

```
web_app_url AND 1=1
```

Επίσης, για λόγους απλότητας, αφού επεξηγήσουμε τη λογική μερικών αιτημάτων και των αντίστοιχων queries που ενσωματώνει το καθένα, θα απλοποιήσουμε τις περιγραφές μας περαιτέρω. Τα αιτήματά μας θα περιγράφονται ως ερωτήσεις προς τον server, ενώ οι πληροφορίες που θα εξάγουμε κάθε φορά θα δίνονται ως απαντήσεις του server, αλλά με ανθρωποκεντρικό τρόπο. Αν αυτό το τελευταίο σας μπέρδεψε, ξεχάστε το. Συνεχίστε να διαβάζετε και θα καταλάβετε τα πάντα.

Know your enemy

Ας κάνουμε μερικούς ακόμα ελέγχους. Αρχικά, θα εξετάσουμε αν είναι αποδεκτά κάποια queries με το SELECT:

```
http://localhost/webapp.php?id=1 AND (SELECT 1)=1
```

Με αυτό το αίτημα, εμφανίζεται πάλι το άρθρο του subZraw. Yeah, baby! Μήπως έχουμε και πρόσβαση σε άλλες βάσεις; Μήπως έχουμε πρόσβαση και στον πίνακα mysql.user; Αυτός ο πίνακας υπάρχει σε κάθε τυπική εγκατάσταση της MySQL. Ε, λοιπόν, αν κατά το στήσιμο του web application είχαμε επιτρέψει στο script να συνδέεται στη βάση μέσω του χρήστη root, η βάση μας θα ήταν τώρα εκτεθειμένη. Προσέξτε το ακόλουθο query:

```
http://localhost/webapp.php?id=1 AND (SELECT 1 FROM mysql.user
LIMIT 0,1)=1
```

Ο χρήστης μέσω του οποίου συνδέεται η εφαρμογή μας στη MySQL δεν έχει πρόσβαση στον πίνακα mysql.user. Έτσι, το παραπάνω αίτημα επέστρεψε το κλασικό μήνυμα λάθους.

Το επόμενο που θα θέλαμε να μάθουμε είναι ποιους πίνακες περιέχει η βάση και τι στήλες περιλαμβάνει ο καθένας. Σημειώστε ότι υπάρχει ένας τρόπος με πολύπλοκα queries που εξετάζει το information_schema της βάσης, και μας επιτρέπει να πληροφορηθούμε αμέσως για τους πινάκες και τις στήλες τους. Αυτή την τεχνική χρησιμοποιούν και τα έτοιμα εργαλεία. Ωστόσο, όταν εργαζόμαστε χειροκίνητα υπάρχει και μια απλούστερη προσέγγιση: Αυτή του να μαντέψουμε τα ονόματα...

```
web_app_url AND (SELECT 1 FROM username LIMIT 0,1)=1
```

Σε ελεύθερη μετάφραση, το παραπάνω query έχει ως εξής: *Υπάρχει πίνακας username;* Το web application απάντησε με Page not found κι επομένως, είναι σαν να μας απάντησε "όχι".

Ας κάνουμε μια ακόμα δοκιμή:

```
web_app_url AND (SELECT 1 FROM users LIMIT 0,1)=1
```

Ελεύθερη μετάφραση: *Υπάρχει πίνακας users;* Το web application απάντησε εμφανίζοντας το άρθρο του subZraw, κι αυτό ισοδυναμεί με "ναι"!

Για να τσεκάρουμε αν υπάρχει μια στήλη (κι όχι πίνακας, πλέον), προσφέρεται μια λίγο πιο παράξενη τεχνική: Θα χρησιμοποιήσουμε τη συνάρτηση concat() για να συνενώσουμε το string "1" με το όνομα που δοκιμάζουμε και στη συνέχεια, με τη συνάρτηση substring(), θα αποκόψουμε τον πρώτο χαρακτήρα. Αν αυτός παραμένει "1", τότε η στήλη που ψάχνουμε υπάρχει.

```
web_app_url AND (SELECT substring(concat(1,login),1,1) FROM users
LIMIT 0,1)=1
```

Ερώτηση: *Υπάρχει η στήλη login;*
Απάντηση: *Όχι φίλε* (εμφανίζεται το μήνυμα "Page not found").

```
web_app_url AND (SELECT substring(concat(1,username),1,1) FROM
users LIMIT 0,1)=1
```

Ερώτηση: *Υπάρχει στήλη username;*
Απάντηση: *Βεβαίως και υπάρχει* (εμφανίζεται το άρθρο του subZraw).

```
web_app_url AND (SELECT substring(concat(1,pass),1,1) FROM users
LIMIT 0,1)=1
```

Ερώτηση: *Υπάρχει στήλη pass;*
Απάντηση: *Φοβάμαι πως όχι* (εμφανίζεται το μήνυμα "Page not found").

```
web_app_url AND (SELECT substring(concat(1,password),1,1) FROM
users LIMIT 0,1)=1
```

Ερώτηση: *Υπάρχει στήλη password;*
Απάντηση: *Μάγος είσαι;* (εμφανίζεται το άρθρο του subZraw).

Με τη βοήθεια της substring() είναι δυνατόν να μάθουμε και την έκδοση της MySQL. Γνωρίζουμε ότι ο αριθμός της έκδοσης είναι της μορφής X.Y, με 3<X<6 και 0<Y<6. Επομένως, ξεκινάμε τις ερωτήσεις μας για το πρώτο ψηφίο της έκδοσης:

```
web_app_url AND substring(@@version,1,1)=4
```

Ερώτηση: *Είναι ο πρώτος χαρακτήρας της version το 4;*
Απάντηση: *Όχι* (εμφανίζεται το μήνυμα "Page not found").

```
web_app_url AND substring(@@version,1,1)=5
```

Ερώτηση: *Είναι ο πρώτος χαρακτήρας της version το 5;*
Απάντηση: *Αμέ* (εμφανίζεται το άρθρο του subZraw).

```
web_app_url AND substring(@@version,3,1)=4
```

Ερώτηση: *Είναι ο τρίτος χαρακτήρας της version το 4;*
Απάντηση: *Μπα...* (εμφανίζεται το μήνυμα "Page not found").

```
web_app_url AND substring(@@version,3,1)=5
```

Ερώτηση: *Είναι ο τρίτος χαρακτήρας της version το 5;*
Απάντηση: *Σωστά!* (εμφανίζεται το άρθρο του subZraw).

Μετά από αυτόν τον κύκλο ερωτήσεων κι απαντήσεων, γνωρίζουμε ότι η εγκατεστημένη έκδοση της MySQL είναι η 5.5. Ζεσταθήκατε; Τώρα αρχίζουν οι ομορφιές! Θα γνωρίσουμε μια τεχνική που ονομάζεται blind ASCII SQL injection.



Ερώτηση: *Πείτε μου, κύριε server, μήπως η MySQL είναι της οικογένειας 4.x;* ***Απάντηση:*** *Όχι καλέ μου άνθρωπε (εμφανίζεται το μήνυμα Page not found). Τι δεν καταλαβαίνεις;*

ASCII stupid question - get a stupid ANSI

Μέχρι στιγμής έχουμε μάθει την έκδοση της βάσης δεδομένων, κάποιους ενδιαφέροντες πίνακες και κάποιες ενδιαφέρουσες στήλες. Ωστόσο, δεν έχουμε ακουμπήσει καθόλου τα πιο σημαντικά δεδομένα, που κείτονται εντός των πινάκων: Τα διάφορα usernames και τα passwords. Αυτά τα στοιχεία αποθηκεύονται σαν character strings. Αυτό ισχύει και για τα συνθηματικά, παρ' όλο που αποθηκεύονται σε κρυπτογραφημένη μορφή. Όμως, τα character strings δεν μπορούν να συνδυαστούν με τις πράξεις Boolean. Τι μπορούμε να κάνουμε για να συνεχίσουμε το παιχνίδι των ερωτήσεων με τον server;

Όπως έχουμε δει, με τη συνάρτηση substring μπορούμε να επιλέγουμε (και να αποκόπτουμε) οποιοδήποτε υποσύνολο χαρακτήρων ενός string. Ακόμα και έναν μεμονωμένο χαρακτήρα. Αυτή η γνώση οδηγεί σε μια βεβιασμένη και δυστυχώς λανθασμένη ιδέα: Στη δημιουργία ερωτημάτων του τύπου "μήπως είναι 's' ο πρώτος χαρακτήρας του πρώτου username;". Δυστυχώς,είναι αδύνατο να εκτελεστεί ένα τέτοιο ερώτημα. Ωστόσο μπορούμε να μετατρέψουμε το ερώτημα σε αριθμητικό. Σ' αυτή την περίπτωση, το query θα λειτουργήσει κανονικά. Για να πετύχουμε αυτή τη μετατροπή, θα χρησιμοποιήσουμε τη συνάρτηση ascii() της MySQL. Η συγκεκριμένη συνάρτηση επιστρέφει τον κωδικό ASCII ενός χαρακτήρα (`http://en.wikipedia.org/wiki/ASCII`). Υποψιάζεστε τι έχουμε βάλει με το

του μας; Αν όχι, δείτε βήμα προς βήμα τι πρόκειται να κάνουμε:

1. Θα κατασκευάσουμε ένα κλασσικό query, στο δεύτερο μέρος του οποίου θα γίνεται αναφορά προς ένα ενδιαφέρον στοιχείο (π.χ., στο username του πρώτου χρήστη).
2. Θα χρησιμοποιήσουμε τη συνάρτηση substring, για να απομονώσουμε τον πρώτο χαρακτήρα από αυτό το "ενδιαφέρον" στοιχείο.
3. Θα χρησιμοποιήσουμε τη ascii(), για να πάρουμε τον κωδικό ASCII που αντιστοιχεί στον πρώτο χαρακτήρα του username.
4. Θα συγκρίνουμε αυτόν τον κωδικό με άλλους, πιθανούς κωδικούς. Όταν μαντέψουμε την τιμή αυτού του κωδικού, θα έχουμε μάθει ποιος είναι ο ASCII κωδικός του πρώτου χαρακτήρα, του username του πρώτου χρήστη.
5. Επαναλαμβάνουμε από το βήμα 2 και μετά, για τον επόμενο χαρακτήρα του username του πρώτου χρήστη.

Νομίζουμε ότι τώρα καταλάβατε τι θέλουμε να κάνουμε, όπως επίσης και το πόσο κοπιαστικό είναι. Ας το δούμε στην πράξη:

```
web_app_url AND ascii(substring((SELECT username FROM users LIMIT
0,1),1,1))=114
```

Ερώτηση: *Έχει ο πρώτος χαρακτήρας του username του πρώτου χρήστη τον ASCII κωδικό 114; Σε απλά ελληνικά, μήπως πρόκειται για τον χαρακτήρα "r";*
Απάντηση: *Όχι φίλε* (εμφανίζεται το μήνυμα "Page not found").

```
web_app_url AND ascii(substring((SELECT username FROM users LIMIT
0,1),1,1))=115
```

Ερώτηση: *Έχει ο πρώτος χαρακτήρας του username του πρώτου χρήστη τον ASCII κωδικό 115; Σε απλά ελληνικά, μήπως πρόκειται για τον χαρακτήρα "s";*
Απάντηση: *Μέσα έπεσες* (εμφανίζεται το άρθρο του subZraw).

Μόλις ανακαλύψαμε τον πρώτο χαρακτήρα του username. Κουράστηκε κανείς; Όχι, ε; Καλά το καταλάβαμε! Βουρ στους επόμενους χαρακτήρες...

Divide and conquer

Εδώ πρέπει να κάνουμε μια παρένθεση και να πούμε ότι στα ερωτήματα με τους κωδικούς ASCII δεν χρησιμοποιείται συνήθως ο τελεστής της ισότητας, αλλά ο ">" (μεγαλύτερο). Αυτό συμβαίνει γιατί ο πίνακας ASCII είναι διατεταγμένος (κάθε χαρακτήρας μετά από κάποιον άλλο έχει μεγαλύτερο κωδικό) και γιατί οι χαρακτήρες των γραμμάτων εμφανίζονται με αλφαβητική σειρά. Έτσι, ο χαρακτήρας "b" έχει μεγαλύτερο κωδικό από το χαρακτήρα "a" και πάει λέγοντας. Το ίδιο ισχύει και για τους κεφαλαίους χαρακτήρες, οι οποίοι παρεμπιπτόντως ξεκινούν αμέσως μετά τους πεζούς. Ας επιστρέψουμε όμως στα queries με τους κωδικούς ASCII. Όταν μαθαίνουμε ότι ο κωδικός ενός χαρακτήρα είναι μεγαλύτερος από εκείνον του "r", ξέρουμε στα σίγουρα ότι ο υπό εξέταση χαρακτήρας δεν μπορεί να είναι "a", "b" , "c", "d", "e", "f", ..., "p", "q" ή "r". Ως εκ τούτου, χρησιμοποιώντας το ">" καταφέρνουμε να γλυτώσουμε πάρα πολλούς ελέγχους! Δείτε δύο παραδείγματα:



Ερώτηση: *Μήπως ο πίνακας users έχει καμία στήλη με το όνομα password;*
Απάντηση: *Μα και βέβαια, φίλε μου (εμφανίζεται το άρθρο του subZraw).*

```
web_app_url AND ascii(substring((SELECT username FROM users LIMIT
0,1),1,1))>109
```

Ερώτηση: *Μήπως ο πρώτος χαρακτήρας του πρώτου username έχει μεγαλύτερο κωδικό από το χαρακτήρα m (109);* Ουσιαστικά, με αυτή την ερώτηση καταφέρνουμε να "διαιρέσουμε" τα διαθέσιμα γράμματα σε δύο μεγάλα τμήματα (a έως m και n έως z).
Απάντηση: *Ναι είναι* (εμφανίζεται το άρθρο του subZraw).

```
web_app_url AND ascii(substring((SELECT username FROM users LIMIT
0,1),1,1))>116
```

Ερώτηση: *Μήπως ο πρώτος χαρακτήρας του πρώτου username έχει μεγαλύτερο κωδικό από το χαρακτήρα "t" (116);* Με αυτή την ερώτηση χωρίζουμε εκ νέου τους εναπομείναντες χαρακτήρες σε δύο ομάδες (n έως t και u έως z).
Απάντηση: *Όχι φίλε* (εμφανίζεται το μήνυμα "Page not found").

Όπως βλέπετε, με δύο μόνο ελέγχους έχουμε αποκλείσει σχεδόν τα τρία τέταρτα των χαρακτήρων. Γνωρίζουμε πλέον ότι ο πρώτος χαρακτήρας του πρώτου username βρίσκεται στο διάστημα [n...t]. Τώρα μπορούμε να συνεχίσουμε με την ίδια μέθοδο ή, εναλλακτικά, να ελέγξουμε συγκεκριμένους χαρακτήρες που εμφανίζονται με μεγαλύτερη συχνότητα:

```
web_app_url AND ascii(substring((SELECT username FROM users LIMIT
0,1),1,1))=115
```

Ερώτηση: *Μήπως ο πρώτος χαρακτήρας είναι το πολύ συχνό "s" (115);*
Απάντηση: *Μπίνγκο, δικέ μου* (εμφανίζεται το άρθρο του subZraw).

Κι άλλα συμπεράσματα

Τώρα που μάθαμε το κολπάκι με τις συγκρίσεις, μπορούμε να συνεχίσουμε το σκάψιμο της βάσης πιο αποδοτικά! Έχουμε μάθει το πρώτο γράμμα του πρώτου username και μας διακατέχει μια συγκρατημένη αισιοδοξία: Θα χρειαστούν ελάχιστοι έλεγχοι, για κάθε χαρακτήρα του username του πρώτου χρήστη. Και κάπως έτσι τίθεται το εξής ερώτημα: Πόσο είναι το μήκος του username του πρώτου χρήστη; Εδώ θα μας βοηθήσει η συνάρτηση char_length():

```
web_app_url AND (SELECT char_length(username) FROM users LIMIT
0,1)>6
```

Ερώτηση: *Μήπως το μήκος του πρώτου username είναι μεγαλύτερο από 6;*
Απάντηση: *Ναι είναι* (εμφανίζεται το άρθρο του subZraw).

```
web_app_url AND (SELECT char_length(username) FROM users LIMIT
0,1)>7
```

Ερώτηση: *Μήπως το μήκος του πρώτου username είναι μεγαλύτερο από 7;*
Απάντηση: *Όχι, δεν είναι* (εμφανίζεται το μήνυμα "Page not found").

Μάλιστα. Εφόσον το μήκος είναι μεγαλύτερο από το 6 αλλά όχι και από το 7, είναι σίγουρα ίσο με 7. Ας δούμε τι γίνεται με το password…

```
web_app_url AND (SELECT char_length(password) FROM users LIMIT
0,1)>8
```

Ερώτηση: *Μήπως το μήκος του password του πρώτου χρήστη, είναι μεγαλύτερο από 8;*
Απάντηση: *Ναι είναι* (εμφανίζεται το άρθρο του subZraw).

```
web_app_url AND (SELECT char_length(password) FROM users LIMIT
0,1)>16
```

Ερώτηση: *Μήπως το μήκος του password του πρώτου χρήστη, είναι μεγαλύτερο από 16;*
Απάντηση: *Ναι είναι* (εμφανίζεται το άρθρο του subZraw).

Αμάν! Λίγο μεγάλο δεν είναι αυτό το password; Μήπως είναι αποθηκευμένο σε κάποια ειδική μορφή;

```
web_app_url AND (SELECT char_length(password) FROM users LIMIT
1,1)>31
```

Ερώτηση: *Μήπως το μήκος του password του πρώτου χρήστη, είναι μεγαλύτερο από 31;*
Απάντηση: *Ναι, είναι* (εμφανίζεται το άρθρο του subZraw).

Αυτό το περιμέναμε, αλλά για να δούμε τι γίνεται και με το 32...

```
web_app_url AND (SELECT char_length(password) FROM users LIMIT
1,1)>32
```

Ερώτηση: *Μήπως το μήκος του password του πρώτου χρήστη, είναι μεγαλύτερο από 32;*
Απάντηση: *Όχι* (εμφανίζεται το μήνυμα "Page not found").

Αχά! Άρα το μήκος του password είναι ακριβώς 32 χαρακτήρες. Η υπόθεση μυρίζει MD5 hashing (32 digits fixed length). Εάν ισχύει κάτι τέτοιο, οι ερωτήσεις που θα κάνουμε για τα password δεν χρειάζεται να σκανάρουν όλη την αλφαβήτα. Εάν τα password είναι αποθηκευμένα σαν MD5 hashes, αρκεί να τσεκάρουμε τα δεκαέξι ψηφία των αριθμών στο δεκαεξαδικό σύστημα (από "0" έως "9" και από "a" έως "f"). Ταυτόχρονα, για αυτούς τους ελέγχους θα μπορούμε να μετατρέπουμε κάθε κεφαλαίο χαρακτήρα σε πεζό. Εφόσον μιλάμε για αριθμούς, το letter case δεν παίζει κανένα ρόλο. Έτσι, μπορούμε να κάνουμε τη δουλειά μας με ακόμα λιγότερες δοκιμές.

Έτοιμα εργαλεία

Ας κάνουμε έναν μικρό απολογισμό και μερικούς ενδιαφέροντες υπολογισμούς. Για τον πρώτο user ισχύουν τα εξής:

- Το username έχει μήκος 7 χαρακτήρες και καθένας από αυτούς

μπορεί να είναι κάποιος από τους 95 (printable) του πίνακα ASCII. Δηλαδή, στη χειρότερη περίπτωση θα χρειαστούν 95 δοκιμές για κάθε χαρακτήρα. Για ολόκληρο το username θα χρειαστούν 630!

- Το password έχει μήκος 32 χαρακτήρες και καθένας από αυτούς μπορεί να είναι οποιοσδήποτε από τα 16 ψηφία των αριθμών στο δεκαεξαδικό. Στη χειρότερη περίπτωση, θα χρειαστούν 16 δοκιμές για κάθε χαρακτήρα του MD5. Επομένως, για ολόκληρο το hash θα χρειαστούν σχεδόν 512 δοκιμές!

Αν σκέφτεστε ότι όλοι αυτοί οι έλεγχοι θα μπορούσαν να γίνουν με το χέρι κι από το browser, είστε τουλάχιστον παρανοϊκοί -- με την καλή έννοια. Αν χρειάζονται 1142 ερωτήματα για τον πρώτο μόνο χρήστη, σκεφτείτε πόσα θα χρειαστούν για όλους. Σε ένα ρεαλιστικό σενάριο, θα ήταν αδύνατο να στείλουμε τόσα HTTP requests χωρίς να μας μπλοκάρει κάποιο IDS ή WAF. Όλο και κάποιο alert θα έφτανε στον admin και το προσωρινό ban της διεύθυνσής μας θα ήταν σχεδόν αναπόφευκτο.

Πώς θα μπορούσαμε να ξεπεράσουμε αυτές τις δυσκολίες; Ας ρίξουμε μερικές ιδέες...

- Να ελαττώσουμε τον αριθμό των δοκιμών και άρα το πλήθος των HTTP requests.
  - ◇ Αυτό μπορεί να γίνει εφαρμόζοντας το κολπάκι που είδαμε νωρίτερα: Διαιρώντας συνέχεια το πλήθος των διαθέσιμων χαρακτήρων σε δύο τμήματα.
  - ◇ Περιορίζοντας το αλφάβητο! Κι αυτό το είδαμε νωρίτερα. Στην περίπτωση των MD5 hashes, δεν χρειάζεται να κάνουμε διάκριση μεταξύ πεζών και κεφαλαίων. Επίσης, μπορούμε και πάλι να εφαρμόσουμε το κολπάκι με τις διαιρέσεις των τμημάτων αν κι όχι τόσο αποδοτικά, αφού τα αριθμητικά ψηφία και τα γράμματα που ενδέχεται να εμφανίζονται σε έναν δεκαεξαδικό αριθμό δεν βρίσκονται σε διαδοχικές θέσεις του πίνακα ASCII.
- Να αυξήσουμε το χρόνο ανάμεσα στα HTTP requests, ώστε να μην κινήσουμε υποψίες. Αυτό μπορεί να γίνει εύκολα από ένα script, το οποίο θα εισάγει μια χρονική καθυστέρηση (ενδεχομένως τυχαία) πριν από κάθε δοκιμή.
- Να αλλάζουμε σε τακτά χρονικά διαστήματα τη διεύθυνση από την οποία πραγματοποιούνται οι δοκιμές, ώστε και πάλι να μην κινούμε υποψίες.

Φαίνεται πως για να κάνουμε τη ζωή μας ευκολότερη, είναι σχεδόν αναγκαστικό να καταφύγουμε σε διάφορα scripts. Πάμε λοιπόν να δούμε το πρώτο από τα δικά μας scripts (retrieve-v1.pl), που βρίσκεται στο γνωστό πακέτο BlindSQL.zip (`http://bit.ly/dh012BlindSQLi`). Πρέπει να πούμε, βέβαια, ότι τα σκριπτάκια μας είναι προσαρμοσμένα στο σενάριο των δοκιμών μας και στοχεύουν αυτόματα στο password του πρώτου χρήστη.



Ερώτηση: *Μήπως ο πρώτος χαρακτήρας του πρώτου username έχει μεγαλύτερο ASCII code από το 115;* ***Απάντηση:*** *Μέσα έπεσες (εμφανίζεται το άρθρο του subZraw).*

Ωστόσο, η λειτουργία τους είναι παραμετρική κι έτσι μπορείτε να μεταβάλλετε τη συμπεριφορά τους. Σε κάθε περίπτωση, ρίχτε μια ματιά στον κώδικα του καθενός και είμαστε βέβαιοι ότι θα καταλάβετε τι πρέπει να κάνετε.

```
alf@nostromo ~/Desktop $ perl retrieve-v1.pl
*** Blind ASCII SQLi by 4li3n - Version 1 (Linear Search) ***
[*] Search started at: Wed Aug 15 19:38:55 EEST 2012
[*] Result for user 0: afc2fdb4aac65655dd99851716327778
[*] Search ended at: Wed Aug 15 19:38:56 EEST 2012
[*] Hits needed: 512
```

Το πρώτο script υπήρξε αποτελεσματικότατο στην αποστολή του. Για το decryption του MD5 hash μπορούμε να χρησιμοποιήσουμε το script από το σχετικό άρθρο του Dr@G (http://deltahacker.gr/?p=6239) ή την απίστευτα πλούσια βάση με hashes (8,7 δισεκατομύρια), στο http://md5decrypter.co.uk. Σημειώστε ότι τα hits που κάνει το script είναι 512, καθώς η αναζήτηση που πραγματοποιεί είναι γραμμική[1]. Αν σας φαίνεται παράξενο το γεγονός ότι τα 512 hits έγιναν στη διάρκεια ενός δευτερολέπτου, σκεφτείτε ότι κάνουμε τις δοκιμαστικές επιθέσεις από το ίδιο μηχάνημα! Σε έναν live στόχο, στο Internet, θα χρειαζόταν πολλή περισσότερη ώρα. Ας δούμε τώρα και το δεύτερο script (retrieve-v2.pl) σε δράση:

```
alf@nostromo ~/Desktop $ perl retrieve-v2.pl
*** Blind ASCII SQLi by 4li3n - Version 2 (Linear Search
Optimized) ***
[*] Search started at: Wed Aug 15 19:39:00 EEST 2012
[*] Result for user 0: afc2fdb4aac65655dd99851716327778
[*] Search ended at: Wed Aug 15 19:39:01 EEST 2012
[*] Hits needed: 251
```

Αυτό το script είναι στοιχειωδώς εξυπνότερο. Όταν βρίσκει έναν χαρακτήρα προχωρά στον επόμενο και δεν μένει στον ίδιο, μέχρι να πραγματοποιηθούν όλες οι πιθανές δοκιμές. Έτσι, τα hits μειώθηκαν στα 251[2]. Ας δούμε τώρα και το τρίτο (retrieve-v3.pl):

```
alf@nostromo ~/Desktop $ perl retrieve-v3.pl
*** Blind ASCII SQLi by 41i3n - Version 3 (Binary Search) ***
[*] Search started at: Wed Aug 15 19:39:04 EEST 2012
[*] Result for user 0: afc2fdb4a0c65650d099851716327778
[*] Search ended at: Wed Aug 15 19:39:04 EEST 2012
[*] Hits needed: 128
```

Πολύ καλύτερα! Αυτή τη φορά χρειάστηκαν μόνο 128 hits. Εδώ χρησιμοποιείται η μέθοδος της δυαδικής αναζήτησης (binary search). Ουσιαστικά, χρησιμοποιείται το κολπάκι με τις διαιρέσεις που αναφέραμε νωρίτερα. Για προφανείς λόγους, αυτή είναι η καλύτερη τεχνική αναζήτησης!

Υπάρχουν πολλοί ακόμα άλλοι τρόποι, με τους οποίους θα μπορούσαν να βελτιστοποιηθούν οι αναζητήσεις. Ο απλούστερος βασίζεται σε μια στατιστική ανάλυση και στη συχνότητα εμφάνισης των χαρακτήρων. Βέβαια αυτή η μέθοδος δεν έχει καμία χρησιμότητα στην περίπτωση ενός MD5

1. Πολυπλοκότητας O(k)·O(n), με k=32 και n=16
2. Η πολυπλοκότητα όμως παραμένει O(k)·O(n), με k=32 και n=16, αφού προκύπτει από το worst case scenario.

hash. Θα μπορούσε όμως να βοηθήσει πολύ στην εύρεση ενός username. Όλα αυτά τα ωραία και πολλά ακόμα, χρησιμοποιούνται από το εργαλείο themole. Εμένα με έχει εντυπωσιάσει με την απλότητα και με την αποτελεσματικότητά του! Σκεφτείτε ότι ζητάει μόνο το URL του τρύπιου web application, καθώς και ένα NEEDLE (δηλαδή ένα matching string, όπως το "by subZraw")[3]. Για να δείτε πόσο εύκολα γίνονται όσα αναλύσαμε, δείτε το μικρό βίντεο που θα βρείτε στο πακέτο BlindSQL.zip. Όσοι χρησιμοποιείτε Windows, θα χρειαστείτε το VLC. Επίσης, έχετε υπόψη σας ότι το themole δέχεται δεκάδες άλλες παραμέτρους, τις οποίες δεν επιδεικνύουμε στο βίντεο. Με αυτές ενεργοποιούνται αντίμετρα, όπως τα IDS evasion filters και διάφορα άλλα.

Τελική αποτίμηση

Όσο αυξάνεται η πολυπλοκότητα των web applications, τόσο αυξάνεται και η πιθανότητα να προκύψει κάποιο injection point, εξαιτίας κάποιας παράλειψης στο input sanitization. Εξάλλου, πέρα από τη μέθοδο επίθεσης που περιγράψαμε αυτή τη φορά, υπάρχουν πολλές ακόμα (όπως, π.χ., union select, error based, time based κ.ά.). Στο κάτω κάτω, τα δεκάδες σχετικά εργαλεία που κυκλοφορούν στο δίκτυο δίνουν ικανοποιητικότατα αποτελέσματα. Έτσι, τα injections διατηρούνται σταθερά στις πρώτες θέσεις προτίμησης των attackers. Πάντως η δική μας προτροπή είναι να μην μπλέκετε με live hosts --ούτε καν για πλάκα-- χωρίς την άδεια του ιδιοκτήτη. Υπάρχει περίπτωση να κάνετε ζημιά, όπως επίσης και να βρείτε το μπελά σας!

```
Terminal
File Edit View Search Terminal Help
alf@nostromo ~/Desktop $ perl retrieve-v1.pl

*** Blind ASCII SQLi by 4li3n - Version 1 (Linear Search) ***

[*] Search started at: Wed Aug 15 19:38:55 EEST 2012
[*] Result for user 0: afc2fdb4aac65655dd99851716327778
[*] Search ended at: Wed Aug 15 19:38:56 EEST 2012
[*] Hits needed: 512

alf@nostromo ~/Desktop $ perl retrieve-v2.pl

*** Blind ASCII SQLi by 4li3n - Version 2 (Linear Search Optimized) ***

[*] Search started at: Wed Aug 15 19:39:00 EEST 2012
[*] Result for user 0: afc2fdb4aac65655dd99851716327778
[*] Search ended at: Wed Aug 15 19:39:01 EEST 2012
[*] Hits needed: 251

alf@nostromo ~/Desktop $ perl retrieve-v3.pl

*** Blind ASCII SQLi by 4li3n - Version 3 (Binary Search) ***

[*] Search started at: Wed Aug 15 19:39:04 EEST 2012
[*] Result for user 0: afc2fdb4a0c65650d099851716327778
[*] Search ended at: Wed Aug 15 19:39:04 EEST 2012
[*] Hits needed: 128

alf@nostromo ~/Desktop $
```

Τα σκριπτάκια μας σε δράση. Το καθένα χρησιμοποιεί διαφορετικό αλγόριθμο αναζήτησης. Το πρώτο είναι μάλλον το πιο χαζό, αφού κάνει τις περισσότερες δοκιμές. Το τρίτο κάνει ελάχιστους ελέγχους και είναι το πιο αποδοτικό! Σε κάθε περίπτωση, η διάρκεια των δοκιμών ήταν ελάχιστη, γιατί οι δοκιμές πραγματοποιούνταν τοπικά...

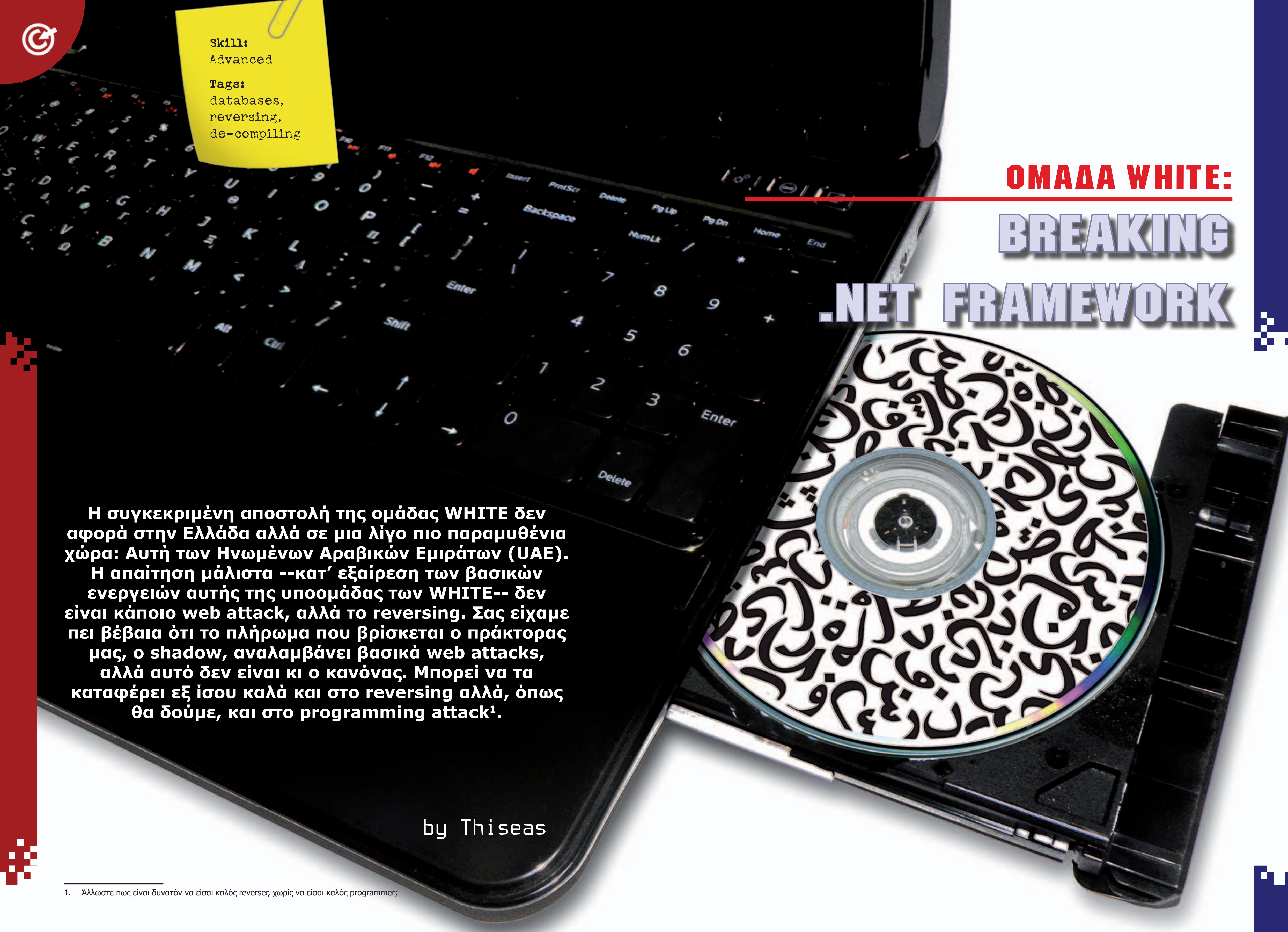
3. Το Themole απαιτεί ένα NEEDLE που να εμφανίζεται όταν η εκάστοτε δοκιμή ήταν επιτυχημένη και το οποίο να επαληθεύεται με τον εντοπισμό κάποιου string. Έτσι, επιλέξαμε το "by subZraw", το οποίο εμφανίζεται όταν μια δοκιμή είναι πετυχημένη κι όχι το "Page not found!", που εμφανίζεται στην αντίθετη περίπτωση.

Skill:
Advanced

Tags:
databases,
reversing,
de-compiling

ΟΜΑΔΑ WHITE:

BREAKING .NET FRAMEWORK



Η συγκεκριμένη αποστολή της ομάδας WHITE δεν αφορά στην Ελλάδα αλλά σε μια λίγο πιο παραμυθένια χώρα: Αυτή των Ηνωμένων Αραβικών Εμιράτων (UAE). Η απαίτηση μάλιστα --κατ' εξαίρεση των βασικών ενεργειών αυτής της υποομάδας των WHITE-- δεν είναι κάποιο web attack, αλλά το reversing. Σας είχαμε πει βέβαια ότι το πλήρωμα που βρίσκεται ο πράκτορας μας, ο shadow, αναλαμβάνει βασικά web attacks, αλλά αυτό δεν είναι κι ο κανόνας. Μπορεί να τα καταφέρει εξ ίσου καλά και στο reversing αλλά, όπως θα δούμε, και στο programming attack[1].

by Thiseas

1. Άλλωστε πως είναι δυνατόν να είσαι καλός reverser, χωρίς να είσαι καλός programmer;

Σε όλες τις αραβόφωνες χώρες, οι τηλεπικοινωνίες έχουν την τιμητική τους. Αυτό το πανδαιμόνιο με τα κινητά τηλέφωνα, τα smartphones και την όποια (κατασκευασμένη) μόδα πηγάζει από αυτά, δεν είναι φαινόμενο μόνο της Ελλάδας αλλά και της Αιγύπτου, της Ιορδανίας, της Σαουδικής Αραβίας κ.ο.κ. Δεν θα μπορούσε λοιπόν να λείπει από όλο αυτόν τον "ανταγωνισμό" και η βιομηχανική κατασκοπία. Με κάτι τέτοιο έχει να κάνει και η συγκεκριμένη αποστολή των WHITE. Συγκεκριμένα, η νέα απαίτηση είναι η ακόλουθη:

> *Μεγάλη εταιρία τηλεπικοινωνιών που δραστηριοποιείται στις Αραβόφωνες χώρες παράγει ένα CD-οδηγό, με καταλόγους από όλες τις επιχειρήσεις της χώρας και τα στοιχεία τους (διευθύνσεις, τηλέφωνα, δραστηριότητα κ.λπ). Αυτό το CD έχει μια βάση δεδομένων που είναι κλειδωμένη και περιέχει όλα τα δεδομένα που εμφανίζονται (και πιθανόν κι άλλα που δεν εμφανίζονται). Καλείστε να ξεκλειδώσετε τη βάση δεδομένων και να εξαγάγετε όλα τα αποθηκευμένα δεδομένα, από όλους τους πίνακές της.*

Πάμε λοιπόν να σας παρουσιάσουμε (με βάση τις πληροφορίες που λάβαμε) τη μέθοδο με την οποία δούλεψε η ομάδα για να φέρει εις πέρας τη συγκεκριμένη αποστολή!

Ομάδα WHITE Section

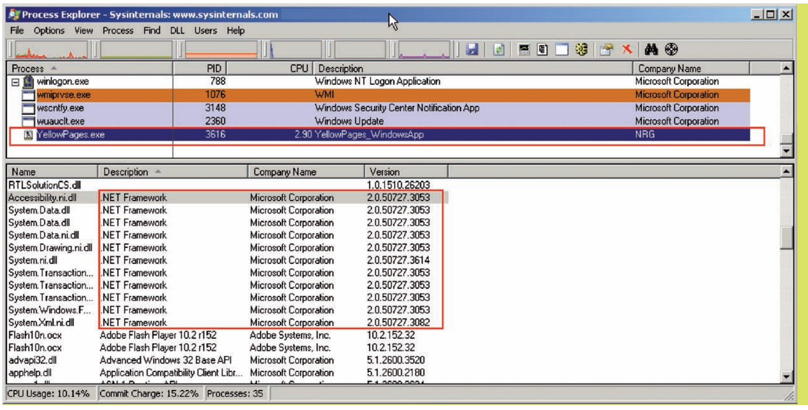
Μας έστειλαν ένα πρόγραμμα που πρέπει να το σπάσουμε και να βρούμε πού κρύβεται η βάση δεδομένων που χρησιμοποιεί. Επίσης θα πρέπει να βρούμε ένα τρόπο για να εξαγάγουμε και όλα τα δεδομένα. Το πρώτο που έχουμε να κάνουμε είναι να τρέξουμε το πρόγραμμα και να το παρακολουθούμε.

Εικόνα 1.
Το πρόγραμμα Yellow Pages, σε πλήρη εξέλιξη.



Στην εικόνα 1 βλέπουμε όλες τις εταιρείες στο Dubai, οι οποίες στην επωνυμία τους περιέχουν τη λέξη "mobile". Απ' ό,τι παρατηρούμε, το πρόγραμμα δεν παρέχει δυνατότητα εξαγωγής όλων των δεδομένων σε κάποιο εξωτερικό αρχείο (π.χ., Excel). Κάποιοι όμως τα χρειάζονται, εξ

ου και η αποστολή μας ;) Το πρόγραμμα τρέχει σε περιβάλλον Windows. Μπορώ λοιπόν να παρακολουθήσω την λειτουργία του με τον Process Explorer (από την Sysinternals, η οποία έχει αγοραστεί από τη Microsoft). Τον τρέχω λοιπόν κι επιλέγω το EXE της εφαρμογής (βλ. εικόνα 2).

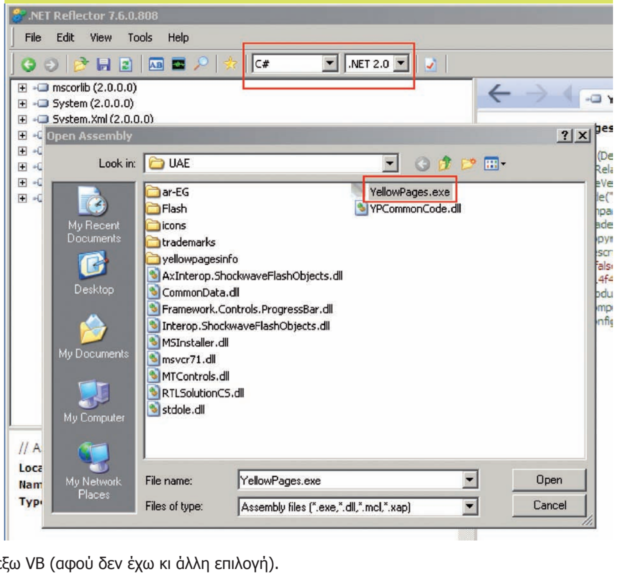


Εικόνα 2. *Το πρόγραμμα-στόχος τρέχει σε .Net framework, έκδοση 2.0.*

Βλέπω ότι το πρόγραμμα τρέχει σε περιβάλλον Microsoft .Net framework 2. Αυτό και μόνο φέρνει στο μυαλό δυό λέξεις: .Net Reflector! Είναι ένας από τους πιο αξιόπιστους de-compilers που έχουν κυκλοφορήσει στην αγορά. Και λέω αγορά, διότι από τις 2 Φεβρουαρίου του 2011 ο Reflector έγινε εμπορικό προϊόν και πωλείται προς 35€. Η έκδοση που χρησιμοποίησα είναι η 7.6, που είναι και η πιο πρόσφατη τη στιγμή αυτή.

Ξεκινάω λοιπόν το Reflector κι επιλέγω το .Net Framework 2, γλώσσα μεταγλώττισης την C#[2] και το πρόγραμμα που θέλω να κάνω de-compile, δηλαδή το yellowpages.exe (βλ. εικόνα 3).

Εικόνα 3. *Decompile με τον Reflector.*



Νομίζω ότι ήρθε πια η ώρα για να εξηγήσω το σχέδιο μου: Κάθε εφαρμογή .Net (και όχι μόνο) που μιλάει με μια βάση δεδομένων, έχει στον κώδικά της μια μεταβλητή μέσω της οποίας πραγματοποιεί τη σύνδεση της με αυτή τη βάση. Η εν λόγω μεταβλητή είναι γνωστή με το όνομα Connection String. Μέσα σε αυτή την μεταβλητή καταχωρούνται πάρα πολύ σημαντικές πληροφορίες, όπως: Το όνομα και η θέση (στο δίσκο) της Βάσης Δεδομένων, το username και το password που χρησιμοποιούνται για να αποκτήσω πρόσβαση στα data, καθώς και διάφορα άλλα στοιχεία που δεν θα μας απασχολήσουν επί του παρόντος. Αν βρω στον κώδικα πού ορίζεται αυτό

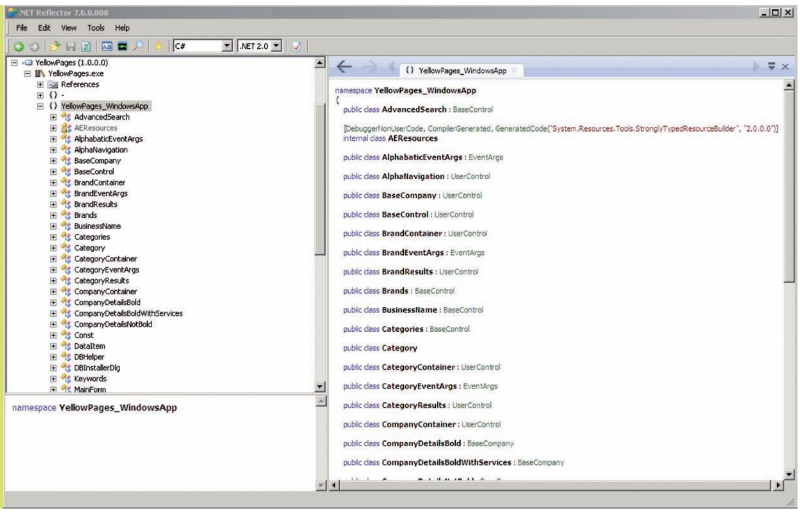
2. Στην τύχη το έβαλα. Αν δεν παίξει κάτι καλά θα επιλέξω VB (αφού δεν έχω κι άλλη επιλογή).

το Connection String, τότε μπορώ εύκολα να φτιάξω ένα πρόγραμμα-κλέφτη το οποίο χρησιμοποιώντας το Connection String που βρήκα από το κανονικό πρόγραμμα θα προσπελάσει την βάση δεδομένων και θα μου εμφανίσει όποια δεδομένα θέλω. Θα χρησιμοποιήσω, λοιπόν, το Reflector, για να βρω αυτό το Connection String.

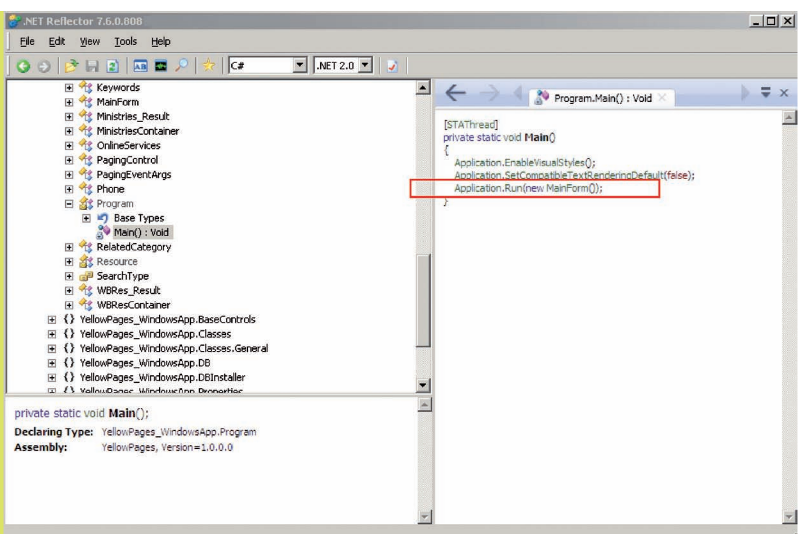
Εντοπισμός του Connection String

Αφού επέλεξα ν' ανοίξω το Yellow Pages, το επόμενο βήμα είναι να προσπαθήσω να παρακολουθήσω την ροή του προγράμματος σαν να έτρεχε κανονικά. Μέσω του Reflector παρατηρώ, σε ιεραρχική δομή, όλα τα namesapaces, τις κλάσεις καθώς και τις member-functions του προς εξέταση προγράμματος (yellowpages.exe, βλ. εικόνα 4).

Εικόνα 4.
Τα βασικά μέρη της εφαρμογής Yellow Pages.

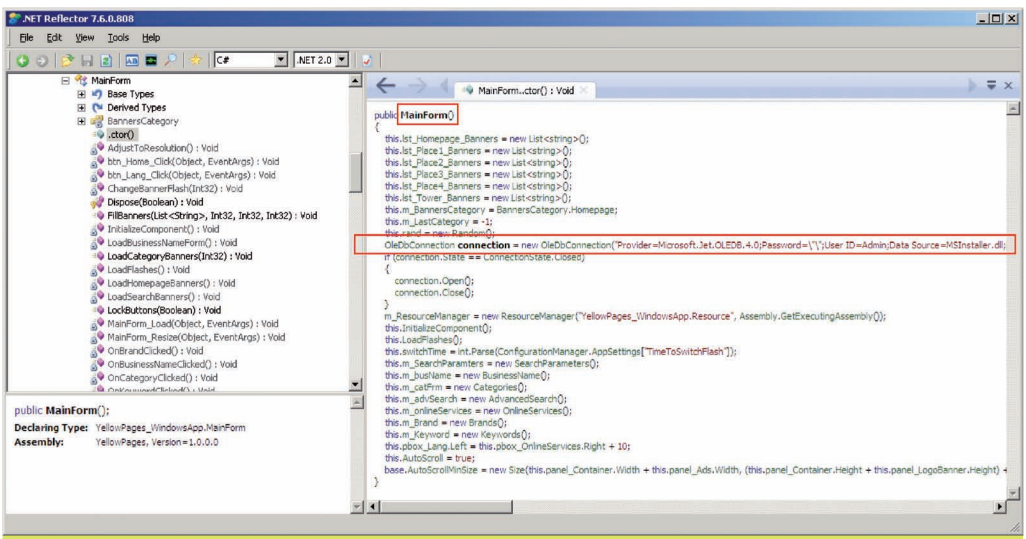


Εικόνα 5.
Η πρώτη και βασική function που εκτελείται, όταν τρέξει το πρόγραμμα - στόχος.



Επάνω στο namespace YellowPages (1.0.0.0) πατάω δεξί click κι επιλέγω Go To Entry Point. Αμέσως, το Reflector με τοποθετεί στη βασική και πρώτη member function που εκτελείται όταν τρέξω το πρόγραμμα, τη main (βλ. εικόνα 5). Πρέπει να πω εδώ ότι έχω σταθεί τυχερός, διότι το πρόγραμμα έχει γραφτεί χωρίς την τεχνική του code obfuscation, χωρίς δηλαδή το σκόπιμο μπέρδεμα του κώδικα με στόχο όταν κάποιος κάνει decompile να μην καταλαβαίνει τίποτα (ή σχεδόν τίποτα). Αυτό το γεγονός έχει το καλό αποτέλεσμα για μένα --και το κακό για την εταιρεία που το έφτιαξε--, ότι μπορώ να δω ακριβώς τον κώδικα που γράφτηκε, με ό,τι αυτό συνεπάγεται.

Παρατηρώντας τη function main βλέπω ότι καλεί (αφού τη δημιουργήσει) την κλάση MainForm. Μπορώ απλά να κάνω click επάνω στην εντολή αυτή (την τελευταία), για να οδηγηθώ στον κώδικα αυτής της κλάσης (βλ. εικόνα 6).



Εικόνα 6: *Εδώ είμαστε: Ιδού το πολυπόθητο Connection String!*

Με την πρώτη! Απ' ό,τι βλέπετε, το Connection String ορίζεται στην πρώτη function του προγράμματος (ως συνήθως), ώστε να μπορεί να χρησιμοποιηθεί αργότερα. Αυτό το πολυπόθητο Connection String είναι το παρακάτω[3]:

```
OleDbConnection connection = new OleDbConnection('Provide
r=Microsoft.Jet.OLEDB.4.0;Password=\'\';User ID=Admin;Data
Source=MSInstaller.dll;Mode=Share Deny Write;Extended
Properties=\'\';Jet OLEDB:System database=\'\';Jet OLEDB:Registry
Path=\'\';Jet OLEDB:Database Password=\'xxxxxxxxxxxxxxxx
xx\';Jet OLEDB:Engine Type=5;Jet OLEDB:Database Locking
Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Global
Bulk Transactions=1;Jet OLEDB:New Database Password=\'\';Jet
OLEDB:Create System Database=False;Jet OLEDB:Encrypt
Database=False;Jet OLEDB:Don't Copy Locale on Compact=False;Jet
OLEDB:Compact Without Replica Repair=False;Jet OLEDB:SFP=False');
```

3. Για ευνόητους λόγους το password έχει καλυφθεί με "xxx...".

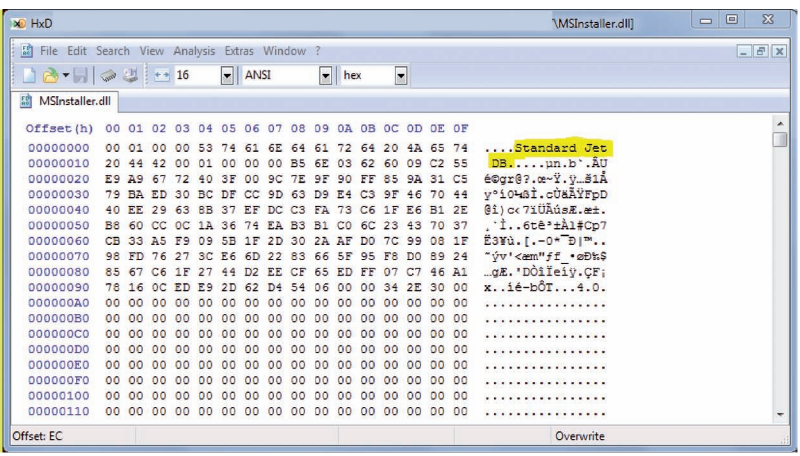
Μπορούμε να εξαγάγουμε κάποια χρήσιμα συμπεράσματα διαβάζοντάς το.

1. Η σύνδεση με τη βάση γίνεται μέσω του OleDB API της Microsoft. Πρόκειται για την επόμενη έκδοση του πολύ πετυχημένου ODBC API. Υλοποιεί το μοντέλο COM (Common Object Model), το οποίο χρησιμοποιούσε η Microsoft κατά κόρον στο παρελθόν. Πλέον, όλα αυτά τείνουν να αντικατασταθούν με το νεότερο και απαλλαγμένο από το COM (επιτέλους!) ADO.Net (ActiveX Data Objects for .Net). Τέλος πάντων, αν και η ίδια η Microsoft δεν έχει αποφασίσει ακόμα ποιο απ' όλα τελικά θέλει να κρατήσει ζωντανό (http://bit.ly/OTTqaC), μπορούμε να πούμε με σιγουριά ότι το OleDB επινοήθηκε για να χρησιμοποιεί κάποιος την SQL σε μη SQL βάσεις δεδομένων (βλ. Microsoft Excel).

2. Η βάση δεδομένων βρίσκεται στον τρέχοντα κατάλογο κι έχει το όνομα MSInstaller.DLL. Αν έδωσαν τέτοιο όνομα οι προγραμματιστές της εφαρμογής για να μπερδέψουν τον υποψήφιο reverser, τότε μάλλον είναι πάρα πολύ… αγνοί! Κάποιος πρέπει να τους μιλήσει σοβαρά για το obfuscation. Παρατηρώντας το συγκεκριμένο αρχείο στο δίσκο, βλέπω ότι έχει μέγεθος 150MB, πράγμα που ενισχύει την άποψη ότι πρόκειται για τη βάση δεδομένων. Με χρήση hex-editor δεν είναι δυνατό να διαβάσω κάποια δεδομένα, γεγονός που δείχνει ότι χρησιμοποιείται κάποιας μορφής encryption. Με τον ίδιο editor φαίνεται καθαρά η υπογραφή Standard Jet DB, της Microsoft database engine (βλ. εικόνα 7).

3. Τα username (Admin) και password (το έχουμε κρύψει για ευνόητους λόγους) φαίνονται καθαρά σε clear text.

Εικόνα 7. *Η υπογραφή στην επικεφαλίδα του αρχείου MSInstaller.DLL προδίδει την αληθινή του φύση! Το παραπλανητικό όνομα αρχείου, μόνο να παραπλανήσει δεν μπορεί ;)*



Δημιουργία του …κλέφτη

Με βάση τα παραπάνω στοιχεία που απέκτησα, μπορώ να κατασκευάσω ένα πρόγραμμα το οποίο θα κάνει χρήση του συγκεκριμένου Connection String και θα διαβάζει τη βάση δεδομένων. Αυτό που στην πραγματικότητα θα φτιάξω θα 'ναι ένας SQL command executer (είναι πολύ βολικός). Δηλαδή, θα υπάρχει μια φόρμα μ' ένα text-box στο οποίο θα βάζει κάποιος την εντολή σε SQL που θέλει να εκτελέσει, θα πατάει ένα button στο στιλ "Execute" και τα δεδομένα από το αποτέλεσμα της εντολής θα εμφανίζονται στη φόρμα. Χμ, υπάρχει όμως ένα προβληματάκι. Πώς στην

ευχή θα εκτελέσει κάποιος μια εντολή SQL, αν δεν γνωρίζει τα ονόματα όλων των πινάκων που υπάρχουν στη βάση δεδομένων;

Προφανώς οφείλω να φτιάξω και μια διαδικασία που να εμφανίζει στο χρήστη όλους τους πίνακες που υπάρχουν στη βάση δεδομένων, έτσι ώστε να μπορέσει αυτός να τους χρησιμοποιήσει στις εντολές του. Το πρόγραμμα θα το φτιάξω σε C#, αν και θα μπορούσα να χρησιμοποιήσω όποια άλλη γλώσσα έχει τη δυνατότητα να διαχειριστεί το OleDB API.

Θα αναλύσω τις δύο παραπάνω διαδικασίες και μετά θα σας δείξω μερικά screenshots από το πρόγραμμα.

Υπάρχει μια συγκεκριμένη διαδικασία μέσω της οποίας --και με χρήση του OleDB API μέσω της συνάρτησης GetOleDbSchemaTable-- μπορεί κάποιος να εξαγάγει τα ονόματα των πινάκων της βάσης δεδομένων. Είναι η παρακάτω.

```
OleDbConnection cn = new OleDbConnection();
DataTable schemaTable;
cn.ConnectionString = "Provider=Microsoft...SFP=False";
cn.Open();
schemaTable = cn.GetOleDbSchemaTable(OleDbSchemaGuid.Tables,
             new Object[] { null, null, null, "TABLE" });
listBox1.Items.Clear();
for (int i = 0; i < schemaTable.Rows.Count; i++)
{
     listBox1.Items.Add(schemaTable.Rows[i].ItemArray[2].
ToString());
}
cn.Close();
```

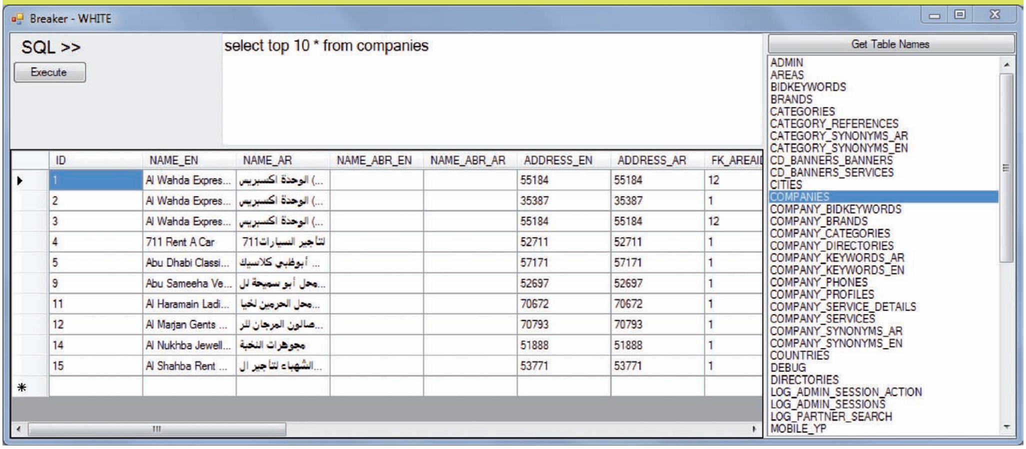
Απ' ό,τι βλέπετε, δεν είναι παραπάνω από 10 γραμμές κώδικα. Μη δώσετε μεγάλη βάση στην 3η γραμμή, διότι την έχω κόψει για να μην πιάνει πολύ χώρο. Το πραγματικό string που περιέχει είναι το Connection String, που βρήκα παραπάνω. Αυτόν τον κώδικα θα τον βάλω μέσα σ' ένα buton-click event, ώστε να ενεργοποιείται με το πάτημα ενός button της φόρμας μου. Το παραπάνω κομμάτι κώδικα "γεμίζει" ένα listbox (με το όνομα listbox1). Αυτό σημαίνει οτι τα αποτελέσματα, δηλαδή τα ονόματα των πινάκων, θα εμφανίζονται σε ένα listbox στη φόρμα μου.

Επίσης, θα υπάρχει (όπως είπα) και το text-box όπου θα μπορεί κάποιος να γράφει εντολές SQL και να τις τρέχει πατώντας ένα button. Ακόμα κι αυτή η διαδικασία είναι πολύ απλή. Ακολουθεί ο κώδικας.

```
string strProvider = 'Provider...SFP=False';
string strSql = textbox_SQL_COMMAND.Text;
OleDbConnection con = new OleDbConnection(strProvider);
OleDbCommand cmd = new OleDbCommand(strSql, con);
con.Open();
cmd.CommandType = CommandType.Text;
OleDbDataAdapter da = new OleDbDataAdapter(cmd);
DataTable dTable = new DataTable();
try
{
     da.Fill(dTable);
     dataGridView1.DataSource = dTable;
}
catch (Exception err)
{
```

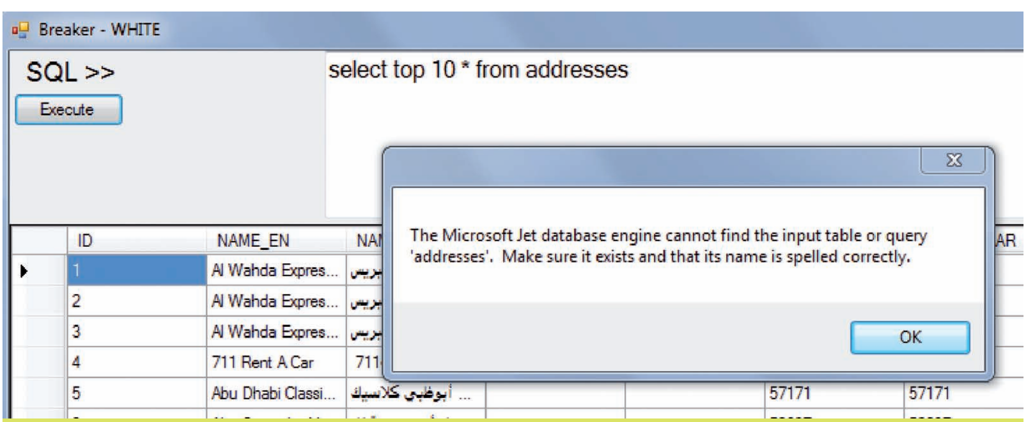
```
        MessageBox.Show(err.Message);
    }
    con.close();
```

Στην 1η γραμμή ορίζω το Connection String (είναι και πάλι σύντμηση αυτού που βρήκα πριν). Στη 2η καταχωρώ σε μια μεταβλητή το περιεχόμενο του text box με την εντολή SQL που έγραψε ο χρήστης. Στις γραμμές 3 και 4 δημιουργώ τις μεταβλητές για το connection και το SQL command. Στην 5, ανοίγω το connection. Στις 6 έως κι 8 φτιάχνω την επικοινωνία του προγράμματος μέσω του OleDB, με τη συγκεκριμένη εντολή SQL που δόθηκε. Οι επόμενες δύο γραμμές (αυτές που είναι μέσα στο βρόχο try, εκτελούν την εντολή που δόθηκε. Απ' ό,τι βλέπετε, εισάγουν τα αποτελέσματα σ' ένα dataGrinView που έχω βάλει στη φόρμα (βλ. εικόνα 8).



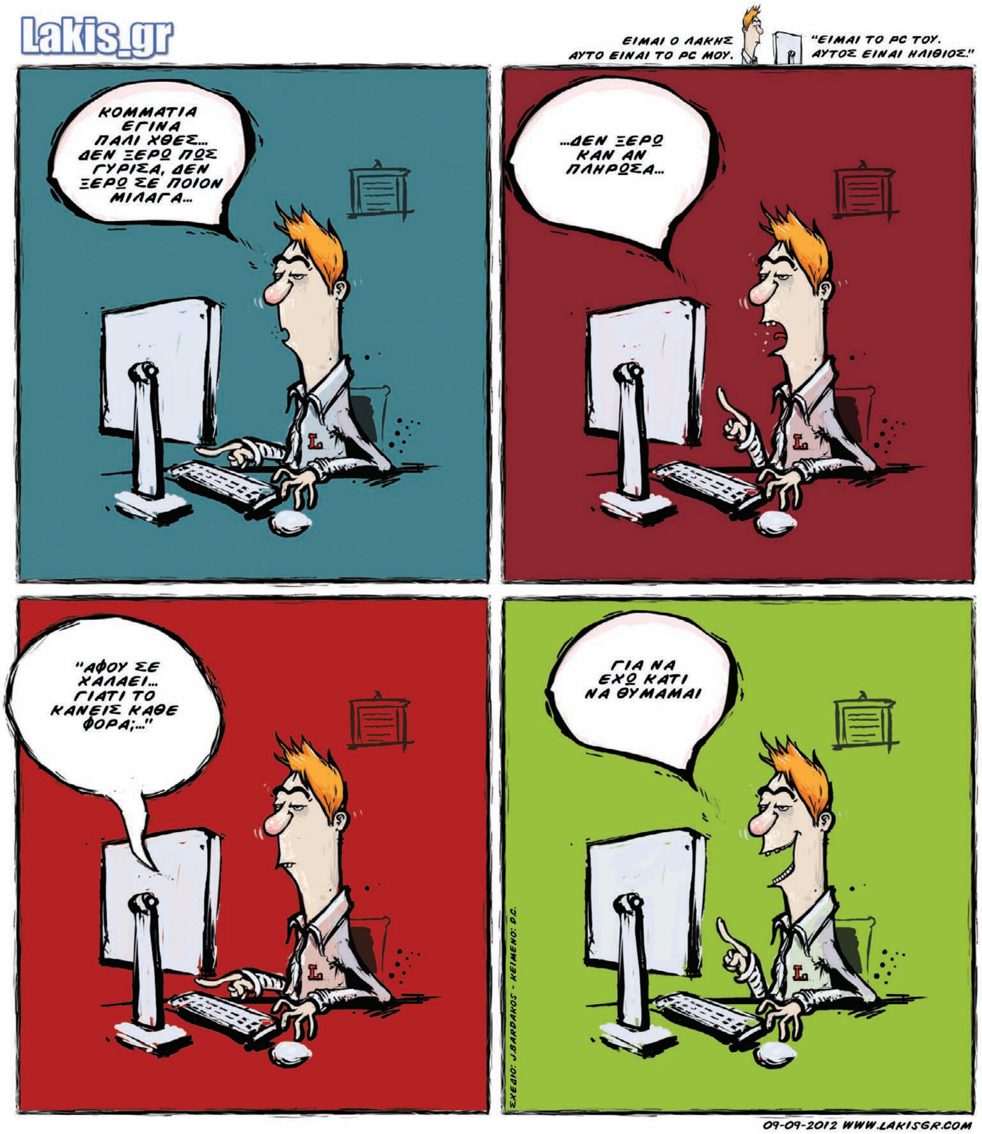
Εικόνα 8. *Ορίστε τα πρώτα δεδομένα, στο πιάτο! Το button GetTableNames, όποτε το πατάω δείχνει, στο Listbox που βρίσκεται ακριβώς από κάτω, όλους τους πίνακες που βρήκε στη βάση.*

Αν κάτι δεν έχει πάει καλά θα εκτελεστεί το exception και θα μου εμφανίσει (μέσω του MessageBox) το μήνυμα λάθους που επέστρεψε ο driver επικοινωνίας με τη βάση δεδομένων (βλ. εικόνα 9). Η τελευταία συνάρτηση, η con.close(), κλείνει το connection ...κι άντε γειά!



Εικόνα 9. *Κάποια λάθη αντιμετωπίζονται εύκολα...*

Ο πίνακας με όλες τις εταιρείες και τα στοιχεία τους, δεν είναι καθόλου μικρός. Όπως με πληροφορεί η SQL, περιέχει 338.348 εγγραφές. Μάλλον αυτός που τον ζήτησε, ήξερε πολύ καλά τι ήθελε...


Lakis_gr
ΕΙΜΑΙ Ο ΛΑΚΗΣ
ΑΥΤΟ ΕΙΝΑΙ ΤΟ PC ΜΟΥ.
"ΕΙΜΑΙ ΤΟ PC ΤΟΥ.
ΑΥΤΟΣ ΕΙΝΑΙ ΗΛΙΘΙΟΣ."
ΚΟΜΜΑΤΙΑ ΕΓΙΝΑ ΠΑΛΙ ΧΘΕΣ... ΔΕΝ ΞΕΡΩ ΠΩΣ ΓΥΡΙΣΑ, ΔΕΝ ΞΕΡΩ ΣΕ ΠΟΙΟΝ ΜΙΛΑΓΑ...
...ΔΕΝ ΞΕΡΩ ΚΑΝ ΑΝ ΠΛΗΡΩΣΑ...
"ΑΦΟΥ ΣΕ ΧΑΛΑΕΙ... ΓΙΑΤΙ ΤΟ ΚΑΝΕΙΣ ΚΑΘΕ ΦΟΡΑ;..."
ΓΙΑ ΝΑ ΕΧΩ ΚΑΤΙ ΝΑ ΘΥΜΑΜΑΙ
ΣΧΕΔΙΟ: J.BARDAKOS - ΚΕΙΜΕΝΟ: D.C.
09-09-2012 WWW.LAKISGR.COM

deltahacker.gr